"Inyección Dependencia (IoC) Anotaciones Con Spring Framework"

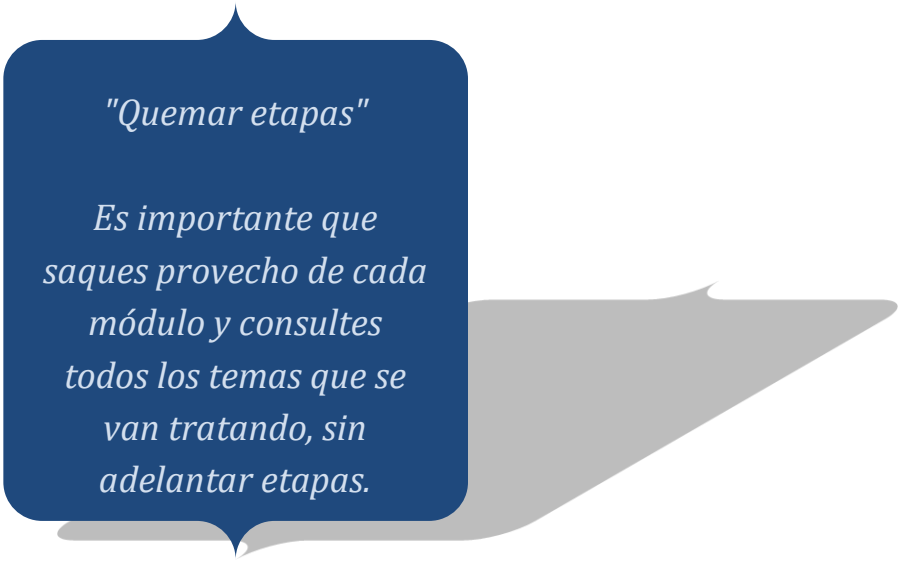
Módulo 2 / 2

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Objetivos

La inyección de dependencias (DI) es la base, el "Core" fundamental de la arquitectura del framework Spring. En este laboratorio profundizaremos más el concepto de inyección de dependencia IoC de Spring con diversos ejemplos de uso de anotaciones. Entre las anotaciones que veremos se incluyen **@Autowired, @Required, @Qualifier, @Inject, @PostConstruct @PreDestory** y **@Resource**.

También veremos la anotación **@Component** con sus derivados estereotipos - **@Service, @Repository** y **@Controller**.



AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Ejercicio 1: Ejemplo "di_anotacion_autowire1_atributo" anotación @Autowired

Anotación @ **Autowired** se utiliza para especificar requisitos DI (en lugar de archivo XML)

Lugares donde puede ser utilizada la anotación @Autowired:

- atributos
- métodos setter
- Constructores
- métodos arbitrarios

1. Clic derecho sobre **di_anotacion_autowire1_atributo->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: 111 Av. Apoquindo Santiago Chile
```

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;

    // Inyectamos el bean direccion
    // usando anotación @Autowired con atributo
    @Autowired
    private Direccion direccion;

    public Direccion getDireccion() {
           return direccion;
    }

    // public void setDireccion(Direccion direccion) {
    // this.direccion = direccion;
    // }

    public String getNombre() {
           return nombre;
    }

    public void setNombre(String nombre) {
           this.nombre = nombre;
    }

    public int getEdad() {
           return edad;
    }
    public void setEdad(int edad) {
           this.edad = edad;
    }
    public float getPeso() {
           return peso;
    }
    public void setPeso(float peso) {
           this.peso = peso;
    }
    public boolean isProgramador() {
           return programador;
    }
    public void setProgramador(boolean esProgramador) {
           this.programador = esProgramador;
    }
}
```

Ejercicio 2: Ejemplo "di_anotacion_autowire2_metodo_setter" @Autowired

1. Clic derecho sobre **di_anotacion_autowire2_metodo_setter->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: 111 Av. Apoquindo Santiago Chile
```

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;
    private Direccion direccion;

    public Direccion getDireccion() {
           return direccion;
    }

    // Inyectamos el bean direccion
    // usando anotación @Autowired con metodo setter
    @Autowired
    public void setDireccion(Direccion direccion) {
           this.direccion = direccion;
    }

    public String getNombre() {
           return nombre;
    }

    public void setNombre(String nombre) {
           this.nombre = nombre;
    }

    public int getEdad() {
           return edad;
    }

    public void setEdad(int edad) {
           this.edad = edad;
    }
    public float getPeso() {
           return peso;
    }
    public void setPeso(float peso) {
           this.peso = peso;
    }
    public boolean isProgramador() {
           return programador;
    }
    public void setProgramador(boolean esProgramador) {
           this.programador = esProgramador;
    }
}
```

Ejercicio 3: Ejemplo "di_anotacion_autowire3_constructor" @Autowired

1. Clic derecho sobre **di_anotacion_autowire3_constructor->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: 111 Av. Apoquindo Santiago Chile
```

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;

public class Persona {
   private String nombre = "Andres Guzman";
   private int edad = 34;
   private float peso = 1.99f;
   private boolean programador = true;
   private Direccion direccion;

   // Inyectamos el bean direccion
   // usando anotación @Autowired en constructor
   @Autowired
   public Persona(Direccion direccion) {
         this.direccion = direccion;
   }

   public Direccion getDireccion() {
         return direccion;
   }

   public void setDireccion(Direccion direccion) {
         this.direccion = direccion;
   }

   public String getNombre() {
         return nombre;
   }
   public void setNombre(String nombre) {
         this.nombre = nombre;
```

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

```
	}

	public int getEdad() {
		return edad;
	}

	public void setEdad(int edad) {
		this.edad = edad;
	}

	public float getPeso() {
		return peso;
	}

	public void setPeso(float peso) {
		this.peso = peso;
	}

	public boolean isProgramador() {
		return programador;
	}

	public void setProgramador(boolean esProgramador) {
		this.programador = esProgramador;
	}
}
```

Ejercicio 4: Ejemplo "di_anotacion_autowire4_metodo_arbitrario" @Autowired

1. Clic derecho sobre **di_anotacion_autowire4_metodo_arbitrario->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: 111 Av. Apoquindo Santiago Chile
```

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;

public class Persona {
   private String nombre = "Andres Guzman";
   private int edad = 34;
   private float peso = 1.99f;
   private boolean programador = true;
   private Direccion direccion;

   // Inyectamos el bean direccion
   // usando anotación @Autowired en otro metodo
   @Autowired
   public void enAlgunArbitrarioMetodo(Direccion direccion) {
         this.direccion = direccion;
   }

   public Direccion getDireccion() {
         return direccion;
   }

   public void setDireccion(Direccion direccion) {
         this.direccion = direccion;
   }

   public String getNombre() {
         return nombre;
   }
```

```
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public int getEdad() {
		return edad;
	}

	public void setEdad(int edad) {
		this.edad = edad;
	}

	public float getPeso() {
		return peso;
	}

	public void setPeso(float peso) {
		this.peso = peso;
	}

	public boolean isProgramador() {
		return programador;
	}

	public void setProgramador(boolean esProgramador) {
		this.programador = esProgramador;
	}

}
```

Ejercicio 5: Ejemplo "di_anotacion_autowire5_setter_requerido" @Autowired

La anotación **@Required** se aplica a los métodos setter o atributos del bean para que sean obligatorio. Se produce una excepción si no se asigna un valor o referencia – de esta forma se puede comprobar si se ha establecido o no el valor/referencia en el atributo, sin necesidad de utilizar código en duro o programación para comprobar, además de establecerlo como requerido y obligatorio.

Al ejecutar el ejemplo observaremos una excepción, ocurre porque el atributo dirección tiene la anotación @Required y además observamos que la anotación @Autowired está comentada. Por lo tanto la referencia del bean dirección nunca se inyectara y al ser requerido lanzará la excepción.

Si des-comentamos la anotación @Autowired, funcionará perfecto.

1. Clic derecho sobre **di_anotacion_autowire5_setter_requerido->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe la excepción ocurrida.

```
Exception in thread "main" org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'persona' defined
resource [beans.xml]: Initialization of bean failed; nested exception is org.springframework.beans.factory.BeanInitializationExcept
'direccion' is required for bean 'persona'
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFac
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFacto
        at org.springframework.beans.factory.support.AbstractBeanFactory$1.getObject(AbstractBeanFactory.java:291)
        at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:22
        at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:288)
        at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:190)
        at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory
        at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContex
        at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:425)
        at org.springframework.context.support.ClassPathXmlApplicationContext.<init>(ClassPathXmlApplicationContext.java:139)
        at org.springframework.context.support.ClassPathXmlApplicationContext.<init>(ClassPathXmlApplicationContext.java:83)
        at com.bolsadeideas.ejemplos.Main.main(Main.java:9)
Caused by: org.springframework.beans.factory.BeanInitializationException: Property 'direccion' is required for bean 'persona'
        at
org.springframework.beans.factory.annotation.RequiredAnnotationBeanPostProcessor.postProcessPropertyValues(RequiredAnnotationBeanPo
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBeanFac
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFac
        ... 11 more
```

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Required;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;
    private Direccion direccion;

    public Direccion getDireccion() {
          return direccion;
    }

    // Usando anotación @Autowired con setter
    //@Autowired
    @Required
    public void setDireccion(Direccion direccion) {
          this.direccion = direccion;
    }

    public String getNombre() {
          return nombre;
    }

    public void setNombre(String nombre) {
          this.nombre = nombre;
    }

    public int getEdad() {
          return edad;
    }

    public void setEdad(int edad) {
          this.edad = edad;
    }

    public float getPeso() {
          return peso;
    }

    public void setPeso(float peso) {
          this.peso = peso;
    }

    public boolean isProgramador() {
          return programador;
    }
```

```
	public void setProgramador(boolean esProgramador) {
		this.programador = esProgramador;
	}

}
```

9. Des-comentamos la anotación **@Autowired** y volvemos a ejecutar la aplicación.

```
package com.bolsadeideas.ejemplos;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Required;

public class Persona {
	private String nombre = "Andres Guzman";
	private int edad = 34;
	private float peso = 1.99f;
	private boolean programador = true;
	private Direccion direccion;

	public Direccion getDireccion() {
		return direccion;
	}

	// Usando anotación @Autowired con setter
	@Autowired
	@Required
	public void setDireccion(Direccion direccion) {
		this.direccion = direccion;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public int getEdad() {
		return edad;
	}

	public void setEdad(int edad) {
		this.edad = edad;
	}

	public float getPeso() {
		return peso;
	}
```

```
	public void setPeso(float peso) {
		this.peso = peso;
	}

	public boolean isProgramador() {
		return programador;
	}

	public void setProgramador(boolean esProgramador) {
		this.programador = esProgramador;
	}

}
```

Ejercicio 6: Ejemplo "di_anotacion_autowire_SpringQualifier1" @Qualifier

Al utilizar autowiring, podríamos tener más de una implementación de beans del mismo tipo, por ejemplos, podríamos tener más de un bean que implemente la misma interface o hereden de la misma clase abstracta, por lo tanto el autowiring nos podría conducir a varios candidatos.

Para evitar conflictos y ambigüedad en que bean inyectar (asociados al mismo tipo) es necesario tener un mayor control sobre el proceso de selección. Una forma de lograr esto es con la anotación **@Qualifier** para clasificar a través de un nombre cuál implementación inyectar.

En el siguiente ejemplo, usaremos una interfaz y dos clases que la implementan, por lo tanto ambas clases pertenecen al mismo tipo de la interface:

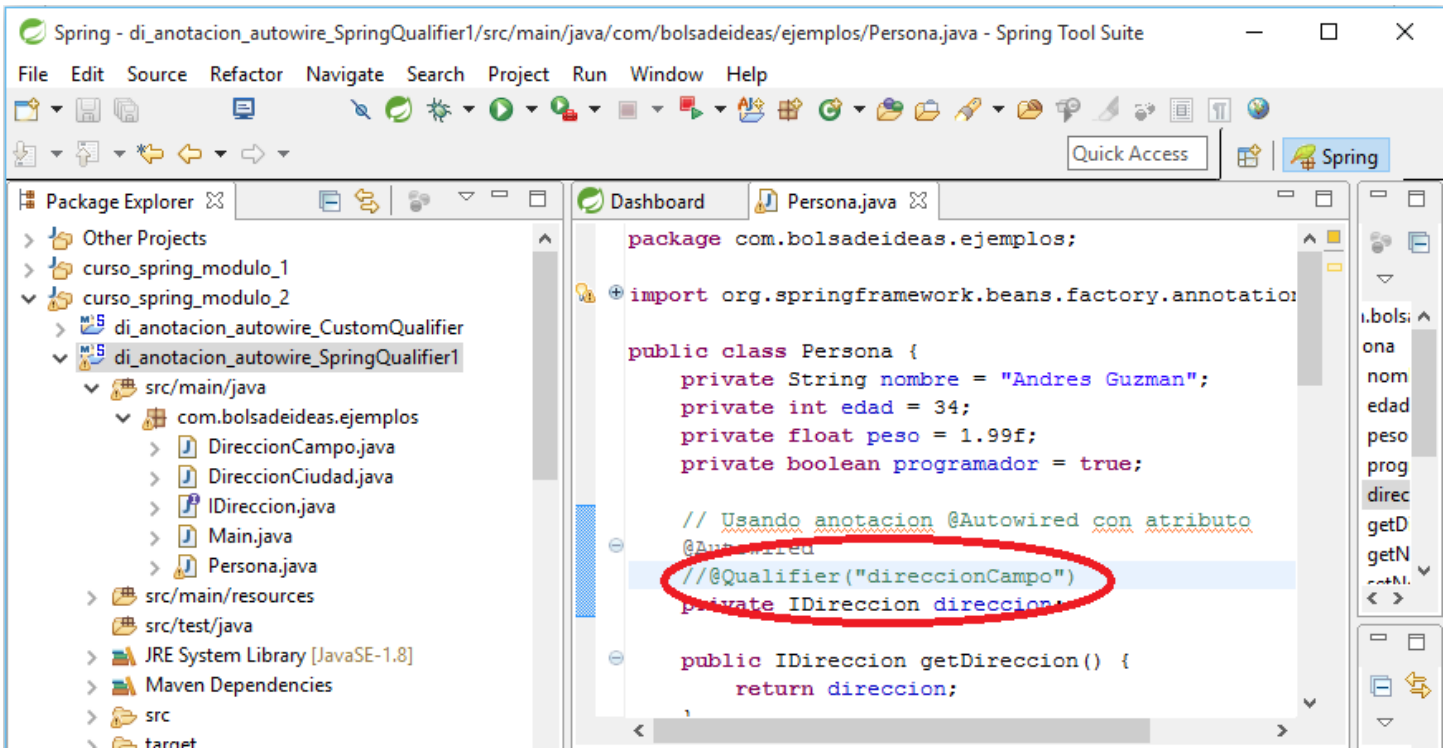
- interface **IDireccion**
- **DireccionCampo** implements IDireccion
- **DireccionCiudad** implements IDireccion

Usaremos la interfaz **IDireccion** como un tipo Java en el autowiring para inyectar la dirección en la clase Persona. Hay dos candidatos posibles de implementación: **DireccionCampo** y **DireccionCiudad**. Por lo tanto debemos ajustar la selección con la anotación **@Qualifier**, de lo contrario lanzará una excepción de ambigüedad, ya que no sabría si inyectar una u otra (DireccionCampo o DireccionCiudad).

1. Clic derecho sobre **di_anotacion_autowire_SpringQualifier1->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Direccion: Direccion: 23 Los Quillayes Pirque Region Metropolitana Chile
```

8. Veamos un experimento, comentemos la anotación **@Qualifier** en Person.java como se muestra a continuación.



9. Volvemos a compilar y ejecutar el ejemplo
10. Observe que aparece una excepción ambigüedad:

```
Exception in thread "main" org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'persona': In
dependencies failed; nested exception is org.springframework.beans.factory.BeanCreationException: Could not autowire field: pri
com.bolsadeideas.ejemplos.IDireccion com.bolsadeideas.ejemplos.Persona.direccion; nested exception is
org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type [com.bolsadeideas.ejemplos.IDirec
single matching bean but found 2: direccionCampo,direccionCiudad
    at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor.postProcessPropertyValues(AutowiredAnnotation
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBeanFac
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFac
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFacto
    at org.springframework.beans.factory.support.AbstractBeanFactory$1.getObject(AbstractBeanFactory.java:305)
    at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:23
    at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:301)
    at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:196)
    at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory
    at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContex
    at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:537)
    at org.springframework.context.support.ClassPathXmlApplicationContext.<init>(ClassPathXmlApplicationContext.java:139)
    at org.springframework.context.support.ClassPathXmlApplicationContext.<init>(ClassPathXmlApplicationContext.java:83)
    at com.bolsadeideas.ejemplos.Main.main(Main.java:10)
Caused by: org.springframework.beans.factory.BeanCreationException: Could not autowire field: private com.bolsadeideas.ejemplos
com.bolsadeideas.ejemplos.Persona.direccion; nested exception is org.springframework.beans.factory.NoUniqueBeanDefinitionExcept
type [com.bolsadeideas.ejemplos.IDireccion] is defined: expected single matching bean but found 2: direccionCampo,direccionCiud
    at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotat
    at org.springframework.beans.factory.annotation.InjectionMetadata.inject(InjectionMetadata.java:88)
    at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor.postProcessPropertyValues(AutowiredAnnotation
    ... 13 more
Caused by: org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type [com.bolsadeideas.ejem
expected single matching bean but found 2: direccionCampo,direccionCiudad
    at org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.java
    at org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.java:9
    at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotat
    ... 15 more
```

11. Eliminamos el comentario y ahora usamos en la anotación **@Qualifier** el valor **direccionCiudad** (en lugar de **direccionCampo**) y volvemos a ejecutar el ejemplo.
12. Observamos que ahora se inyecta la referencia del bean **direccionCiudad**:

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Direccion: Direccion: 111 Av. Apoquindo Santiago Chile
```

13. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;

    // Usando anotación @Autowired con atributo
    @Autowired
    @Qualifier("direccionCampo")
    private IDireccion direccion;

    public IDireccion getDireccion() {
           return direccion;
    }

    // public void setDireccion(IDireccion direccion) {
    // this.direccion = direccion;
    // }

    public String getNombre() {
           return nombre;
    }

    public void setNombre(String nombre) {
           this.nombre = nombre;
    }

    public int getEdad() {
           return edad;
    }

    public void setEdad(int edad) {
           this.edad = edad;
    }

    public float getPeso() {
           return peso;
    }

    public void setPeso(float peso) {
           this.peso = peso;
    }

    public boolean isProgramador() {
           return programador;
    }
```

```
    public void setProgramador(boolean esProgramador) {
          this.programador = esProgramador;
    }

}
```

14. Abrir y estudiar la interface **IDireccion.java**.

```
package com.bolsadeideas.ejemplos;
public interface IDireccion {
    String getDireccionCompleta();
}
```

15. Abrir y estudiar la clase **DireccionCiudad.java**.

```
package com.bolsadeideas.ejemplos;

public class DireccionCiudad implements IDireccion {

    private int numero = 111;
    private String calle = "Av. Apoquindo";
    private String ciudad = "Santiago";
    private String pais = "Chile";

    public int getNumero() {
          return numero;
    }

    public void setNumero(int numero) {
          this.numero = numero;
    }

    public String getCalle() {
          return calle;
    }

    public void setCalle(String calle) {
          this.calle = calle;
    }

    public String getCiudad() {
          return ciudad;
    }

    public void setCiudad(String ciudad) {
          this.ciudad = ciudad;
    }
```

```
	public String getPais() {
		return pais;
	}

	public void setPais(String pais) {
		this.pais = pais;
	}

	@Override
	public String getDireccionCompleta() {
		String direccionCompleta = "Dirección: " + getNumero() + " "
				+ getCalle() + " " + getCiudad() + " " + getPais();
		return direccionCompleta;
	}

}
```

16. Abrir y estudiar la clase **DireccionCampo.java**.

```
package com.bolsadeideas.ejemplos;
public class DireccionCampo implements IDireccion {

	private int numero = 23;
	private String parcela = "Los Quillayes";
	private String comuna = "Pirque";
	private String region = "Region Metropolitana";
	private String pais = "Chile";

	public int getNumero() {
		return numero;
	}

	public void setNumero(int numero) {
		this.numero = numero;
	}

	public String getParcela() {
		return parcela;
	}

	public void setParcela(String parcela) {
		this.parcela = parcela;
	}

	public String getComuna() {
		return comuna;
	}
```

```
	public void setComuna(String comuna) {
		this.comuna = comuna;
	}

	public String getRegion() {
		return region;
	}

	public void setRegion(String region) {
		this.region = region;
	}

	public String getPais() {
		return pais;
	}

	public void setPais(String pais) {
		this.pais = pais;
	}

	@Override
	public String getDireccionCompleta() {
		String direccionCompleta = "Dirección: " + getNumero() + " "
				+ getParcela() + " " + getComuna() + " " + getRegion() + " "
				+ getPais();
		return direccionCompleta;
	}

}
```

17. Abrir y estudiar archivo XML **bean.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"
xmlns:context="http://www.springframework.org/schema/context"
   xsi:schemaLocation="
         http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context.xsd">

   <!-- habilitamos el uso de anotaciones -->
   <context:annotation-config />

   <!-- declaramos bean "persona" -->
   <bean id="persona" class="com.bolsadeideas.ejemplos.Persona" />

   <!-- declaramos bean "direccion" que será inyectado en persona, usando anotación
         @Autowired con atributo -->
   <bean id="direccionCampo" class="com.bolsadeideas.ejemplos.DireccionCampo" />
   <bean id="direccionCiudad" class="com.bolsadeideas.ejemplos.DireccionCiudad" />

</beans>
```

Ejercicio 7: Ejemplo "di_anotacion_autowire_SpringQualifier2" @Qualifier

El valor o referencia que se inyectará al atributo dependerá del valor en la anotación @Qualifier, el cual se define en el archivo XML de configuración (del contexto de spring), anidado en el elemento <bean> como subelemento <qualifier value="abc">

1. Clic derecho sobre **di_anotacion_autowire_SpringQualifier2->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: Dirección: 23 Los Quillayes Pirque Region Metropolitana Chile
```

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;

    // Usando anotación @Autowired con atributo
    @Autowired
    @Qualifier("campo")
    private IDireccion direccion;

    public IDireccion getDireccion() {
           return direccion;
    }

    // public void setDireccion(IDireccion direccion) {
    // this.direccion = direccion;
    // }

    public String getNombre() {
           return nombre;
    }

    public void setNombre(String nombre) {
           this.nombre = nombre;
    }

    public int getEdad() {
           return edad;
    }

    public void setEdad(int edad) {
           this.edad = edad;
    }

    public float getPeso() {
           return peso;
    }

    public void setPeso(float peso) {
           this.peso = peso;
    }

    public boolean isProgramador() {
           return programador;
    }
```

```
    public void setProgramador(boolean esProgramador) {
          this.programador = esProgramador;
    }

}
```

9. Abrir y estudiar archivo XML **bean.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"
xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
          http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
          http://www.springframework.org/schema/context
          http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- habilitamos el uso de anotaciones -->
    <context:annotation-config />

    <!-- declaramos bean "persona" -->
    <bean id="persona" class="com.bolsadeideas.ejemplos.Persona" />

    <!-- declaramos bean "direccion" que será inyectado en persona, usando anotación
          @Autowired con atributo -->
    <bean id="direccionCampo" class="com.bolsadeideas.ejemplos.DireccionCampo">
          <qualifier value="campo" />
    </bean>
    <bean id="direccionCiudad" class="com.bolsadeideas.ejemplos.DireccionCiudad">
          <qualifier value="ciudad" />
    </bean>

</beans>
```

Ejercicio 8: Ejemplo "di_anotacion_autowire_CustomQualifier"

En el siguiente ejemplo vamos a crear nuestra propia anotación clasificadora (custom qualifier annotations). Simplemente definimos una anotación y proveemos la anotación @Qualifier dentro de nuestra definición.

1. Clic derecho sobre **di_anotacion_autowire_CustomQualifier->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: Dirección: 111 Av. Apoquindo Santiago Chile
```

8. Abrir y estudiar la anotación **NuestroQualifier.java**. (nuestra anotación creada a medida)

```
package com.bolsadeideas.ejemplos;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

import org.springframework.beans.factory.annotation.Qualifier;

// Creamos nuestra propia anotacion qualifier llamada NuestroQualifier
@Target({ ElementType.FIELD, ElementType.PARAMETER })
@Retention(RetentionPolicy.RUNTIME)
@Qualifier
public @interface NuestroQualifier {
   String value();
}
```

9. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;

    // Usando anotación @Autowired con atributo
    @Autowired
    @NuestroQualifier("ciudad")
    private IDireccion direccion;

    public IDireccion getDireccion() {
           return direccion;
    }

    // public void setDireccion(IDireccion direccion) {
    // this.direccion = direccion;
    // }

    public String getNombre() {
           return nombre;
    }

    public void setNombre(String nombre) {
           this.nombre = nombre;
    }

    public int getEdad() {
           return edad;
    }

    public void setEdad(int edad) {
           this.edad = edad;
    }

    public float getPeso() {
           return peso;
    }

    public void setPeso(float peso) {
           this.peso = peso;
    }

    public boolean isProgramador() {
           return programador;
    }
```

```
	public void setProgramador(boolean esProgramador) {
		this.programador = esProgramador;
	}

}
```

10. Abrir y estudiar archivo XML **bean.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"
xmlns:context="http://www.springframework.org/schema/context"
	xsi:schemaLocation="
		http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/context
		http://www.springframework.org/schema/context/spring-context.xsd">

	<!-- habilitamos el uso de anotaciones -->
	<context:annotation-config />

	<!-- declaramos bean "persona" -->
	<bean id="persona" class="com.bolsadeideas.ejemplos.Persona" />

	<!-- declaramos bean "direccion" que será inyectado en persona, usando anotación
		@Autowired con atributo -->
	<bean id="direccionCampo" class="com.bolsadeideas.ejemplos.DireccionCampo">
		<qualifier type="NuestroQualifier" value="campo" />
	</bean>
	<bean id="direccionCiudad" class="com.bolsadeideas.ejemplos.DireccionCiudad">
		<qualifier type="NuestroQualifier" value="ciudad" />
	</bean>

</beans>
```

Ejercicio 9: Ejemplo "di_anotacion_Inject " usando la anotación @Inject

La anotación **@Inject** (JSR 330) es propia de la tecnología EJB de Oracle, sin embargo puede ser utilizada en Spring en lugar de @Autowired.

1. Clic derecho sobre **di_anotacion_Inject->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: Dirección: 111 Av. Apoquindo Santiago Chile
```

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import javax.inject.Inject;

import org.springframework.beans.factory.annotation.Qualifier;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;

    // Usamos @JSR 330 @Inject en vez de @Autowired
    //@Autowired
    @Inject
    @Qualifier("direccionCiudad")
    private IDireccion direccion;

    public IDireccion getDireccion() {
          return direccion;
    }

    // public void setDireccion(IDireccion direccion) {
    // this.direccion = direccion;
    // }

    public String getNombre() {
          return nombre;
    }

    public void setNombre(String nombre) {
          this.nombre = nombre;
    }

    public int getEdad() {
          return edad;
    }

    public void setEdad(int edad) {
          this.edad = edad;
    }

    public float getPeso() {
          return peso;
    }

    public void setPeso(float peso) {
          this.peso = peso;
    }
```

```
	public boolean isProgramador() {
		return programador;
	}

	public void setProgramador(boolean esProgramador) {
		this.programador = esProgramador;
	}

}
```

9. Abrir y estudiar archivo XML **bean.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"
xmlns:context="http://www.springframework.org/schema/context"
	xsi:schemaLocation="
		http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/context
		http://www.springframework.org/schema/context/spring-context.xsd">

	<!-- habilitamos el uso de anotaciones -->
	<context:annotation-config />

	<!-- declaramos bean "persona" -->
	<bean id="persona" class="com.bolsadeideas.ejemplos.Persona" />

	<!-- declaramos bean "direccion" que será inyectado en persona, usando anotación
		@Autowired con atributo -->
	<bean id="direccionCampo" class="com.bolsadeideas.ejemplos.DireccionCampo" />
	<bean id="direccionCiudad" class="com.bolsadeideas.ejemplos.DireccionCiudad" />

</beans>
```

Ejercicio 9: Ejemplo "di_anotacion_PostConstruct"

Las anotaciones **@PostConstruct** y **@PreDestroy** ofrecen una alternativa para manejar eventos al inicializar un objeto y antes de su destrucción.

1. Clic derecho sobre **di_anotacion_PostConstruct->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
--->Ciudad Anterior = Santiago, Nueva Ciudad = Viña del Mar
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: 111 Av. Apoquindo Viña del Mar Chile
```

8. Abrir y estudiar la clase **Direccion.java**.

```
package com.bolsadeideas.ejemplos;
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;

public class Direccion {

   private int numero = 111;
   private String calle = "Av. Apoquindo";
   private String ciudad = "Santiago";
   private String pais = "Chile";

   public int getNumero() {
         return numero;
   }

   public void setNumero(int numero) {
         this.numero = numero;
   }

   public String getCalle() {
         return calle;
   }

   public void setCalle(String calle) {
         this.calle = calle;
   }

   public String getCiudad() {
         return ciudad;
   }

   public void setCiudad(String ciudad) {
         this.ciudad = ciudad;
   }

   public String getPais() {
         return pais;
   }

   public void setPais(String pais) {
         this.pais = pais;
   }

   @PostConstruct
   public void hacerAlgoDespuesConstruct() {
         String ciudadAnterior = ciudad;
         ciudad = "Viña del Mar";
         System.out.println("--->Ciudad Anterior = " + ciudadAnterior
                     + ", Nueva Ciudad = " + ciudad);
   }
```

```
    @PreDestroy
    public void hacerAlgoAntesDestroy() {
          System.out
                      .println("--->Hacer algo antes de que se destruya el objeto
Direccion!!!");
    }
}
```

Ejercicio 10: Ejemplo "di_anotacion_Resource" usando la anotación @Resource

Spring también soporta inyección usando la anotación JSR-250 @Resource sobre atributos o métodos setter. Esta anotación es propia de la tecnología Java EE6 (ej: en JSF 2 y managed beans, y EJB).
@Resource toma un atributo 'name', que por defecto Spring interpreta este valor como el nombre del bean para inyectar. En otras palabras hace lo mismo que las anotacines @Autowired y @Qualifier("nombreBean") pero resumida en una sola: @Resource("nombreBean ").

1. Clic derecho sobre **di_anotacion_Resource->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
--->Ciudad Anterior = Santiago, Nueva Ciudad = Viña del Mar
Nombre:Andres Guzman
Edad:34
Peso: 1.99
Es Programador?: true
Dirección: 111 Av. Apoquindo Viña del Mar Chile
```

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

8. Abrir y estudiar la clase **Persona.java**.

```
package com.bolsadeideas.ejemplos;
import javax.annotation.Resource;

public class Persona {
    private String nombre = "Andres Guzman";
    private int edad = 34;
    private float peso = 1.99f;
    private boolean programador = true;

    //@Autowired
    private Direccion direccion;

    public Direccion getDireccion() {
          return direccion;
    }

    @Resource(name="miDireccion")
    public void setDireccion(Direccion direccion) {
    this.direccion = direccion;
    }

    public String getNombre() {
          return nombre;
    }

    public void setNombre(String nombre) {
          this.nombre = nombre;
    }

    public int getEdad() {
          return edad;
    }

    public void setEdad(int edad) {
          this.edad = edad;
    }

    public float getPeso() {
          return peso;
    }

    public void setPeso(float peso) {
          this.peso = peso;
    }

    public boolean isProgramador() {
          return programador;
    }
```

```
	public void setProgramador(boolean esProgramador) {
		this.programador = esProgramador;
	}

}
```

9. Abrir y estudiar archivo XML **bean.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"
xmlns:context="http://www.springframework.org/schema/context"
	xsi:schemaLocation="
		http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/context
		http://www.springframework.org/schema/context/spring-context.xsd">

	<!-- habilitamos el uso de anotaciones -->
	<context:annotation-config />

	<!-- declaramos bean "persona" -->
	<bean id="persona" class="com.bolsadeideas.ejemplos.Persona" />

	<!-- declaramos bean "direccion" que será inyectado en persona, usando anotación
		@Resource con atributo -->
	<bean id="miDireccion" class="com.bolsadeideas.ejemplos.Direccion" />

</beans>
```

Ejercicio 11: Ejemplo "di_Service_Anotacion" usando la anotación @Service

Podemos anotar las clases Service con la anotación @Service y las clases Dao/Repository con @Repository. Ambas anotaciones @Service y @Repository son estereotipos (sub tipos de) @Component.

Particularmente @Repository a es una anotación de Spring que la usamos para marcar a la clase DAO/Repository como un Component de Spring, indicando que esta es una clase relacionada con la capa de persistencia, por lo tanto cualquier error que ocurra se traducirá en DataAccessException org.springframework.dao.DataAccessException. A partir de Spring de 2.5, esta anotación sirve también como una especialización o tipo de @Componente, también son auto detectadas a través de scann de componentes.

Además @Repository indica que debe ser un Singleton (sólo habrá una instancia de la clase DAO, y todos los Threads de la aplicación la compartirán).

1. Clic derecho sobre **di_Service_Anotacion->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Hola, Bruce Lee
```

8. Abrir y estudiar la interfaz **ClienteDao.java** y la clase **ClienteDaoImpl.java**.

```
package com.bolsadeideas.ejemplos;
public interface ClienteDao {
   public String getDetalleCliente();
}
```

```
package com.bolsadeideas.ejemplos;
import org.springframework.stereotype.Repository;

/*
 * @Repository Esta es una anotación de Spring es para marcar
 * a la clase DAO como un Component de Spring, indicando que esta es una
 * clase relacionada con la capa de persistencia, por lo tanto
 * cualquier error que ocurra se traducirá en DataAccessException
 * org.springframework.dao.DataAccessException.
 *
 * A partir de Spring de 2.5, esta anotación sirve también
 * como una especialización o tipo de @Componente, también
 * son auto detectadas a través de scann de componentes.
 *
 * Además @Repository indica que debe ser un Singleton
 * (sólo habrá una instancia de la clase ExpedienteDAO ,
 * y todos los Threads de la aplicación la compartirán).
 */
@Repository("clienteDao")
public class ClienteDaoImpl implements ClienteDao {

   public String getDetalleCliente() {
          // Para simplicidad del ejemplo, retornamos un String con el nombre.
          return "Bruce Lee";
   }
}
```

9. Abrir y estudiar la interfaz **ClienteService.java** y la clase **ClienteServiceImpl.java**.

```
package com.bolsadeideas.ejemplos;
public interface ClienteService {
   public String getSaludoCliente();
}
```

```
package com.bolsadeideas.ejemplos;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.bolsadeideas.ejemplos.dao.ClienteDao;

@Service("clienteService")
public class ClienteServiceImpl implements ClienteService {

   @Autowired
   ClienteDao clienteDao;

   public String getSaludoCliente() {
         String saludo = "Hola, " + clienteDao.getDetalleCliente();
         return saludo;
   }

}
```

10. Abrir y estudiar la clase **Main.java**.

```
package com.bolsadeideas.ejemplos;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.bolsadeideas.ejemplos.services.ClienteService;

public class Main {
   public static void main(String[] args) {
         ApplicationContext context = new ClassPathXmlApplicationContext(
                     new String[] { "beans.xml" });

         ClienteService clienteService = (ClienteService) context
                     .getBean("clienteService");
         System.out.println(clienteService.getSaludoCliente());

   }
}
```

11. Abrir y estudiar archivo XML **bean.xml**.

Cualquier clases que estén anotadas con @Component y sus estereotipos (@Service, @Repository, @Controller) será automáticamente escanea y detecta - por lo tanto no hay necesidad de declararlos en el archivo 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"
xmlns:context="http://www.springframework.org/schema/context"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

   <context:component-scan base-package="com.bolsadeideas.ejemplos" />

</beans>
```

Ejercicio 12: Ejemplo "DI Robot" usando la anotación @

1. Clic derecho sobre **di_robot_anotaciones->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Ejecutamos la aplicación: **Run As->AspectJ/Java Application**.
5. Seleccionamos la clase que contiene el método **main**.
6. Observe el resultado.

```
Moviendo la cabeza
Moviendo el brazo : derecho
Moviendo el brazo : izquierdo
Corriendo con la pierna : derecha
Corriendo con la pierna : izquierda
```

7. Como **tarea** y ejercicio estudiar todo el proyecto, clases anotadas y archivo de configuración bean.xml
8. Cualquier duda lo revisamos en el foro.

Ejercicio 13: Ejemplo "DI Robot" usando XML

1. Clic derecho sobre **di_robot_xml->Run As**
2. Ejecutamos "**Maven clean**".
3. Ejecutamos "**Maven install**".
4. Hacemos un **Maven->Update Project…**
5. Ejecutamos la aplicación: **Run As->Java Application**.
6. Seleccionamos la clase que contiene el método **main**.
7. Observe el resultado.

```
Moviendo la cabeza
Moviendo el brazo : derecho
Moviendo el brazo : izquierdo
Corriendo con la pierna : derecha
Corriendo con la pierna : izquierda
```

8. Como **tarea** y ejercicio estudiar todo el proyecto, clases y archivo de configuración bean.xml
9. Cualquier duda lo revisamos en el foro.

Resumen

En este laboratorio abarcamos más el concepto y uso de anotaciones en el patrón **Inversión de Control** de Spring.

Fin.

Envía tus consultas a los foros!

Aquí es cuando debes sacarte todas las dudas haciendo consultas en los foros correspondientes

Nunca subestimes los ejercicios y toma un tiempo prudencial para empezar a trabajar (no dejes nada para último momento).

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com