Injecting Sub Types into Super Types.

Lab21: Files required

1. Lab21.java	Same as Lab19
2. A.java	Same as Lab19
3. B.java	Same as Lab19
4. CustomerDAO.java	Same as Lab19
5. CustomerDAOImpl.java	Same as Lab19
6. Hello.java	Same as Lab19
7. JLCAppConfig.java	Updated in Lab21

7. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowire;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="aobj")
	public A createA() {
		return new A();
	}

	@Bean(name="mybo")
	public B createB() {
		return new B();
	}

	@Bean(name="customerDAO")
	public CustomerDAO getCustDAO() {
		return new CustomerDAOImpl();
	}

	@Bean(name="myhello",autowire = Autowire.BY_NAME)
	public Hello createHello() { //AutoWiring
		return new Hello();
	}
}
```

Annotation based AutoWiring:

- Field Injection can be implemented with Annotation based AutoWiring
- Field Injection :
  - Injecting the Bean Dependencies directly without any Setter methods or Constructor is called Field Injection.

Using @Autowired:

- @Autowired can be used in two ways.
  1. ByType autowire process
  2. ByName autowire process

1) ByType autowiring with @Autowired

- When you use @Autowired, then by default, beans will be detected based on byType autowire process and inject them directly without any Setter methods or Constructor.

A) @Autowired(required=true) / @Autowired

Case 1: What happens when 0 Matching Beans found.(Refer Lab22)

- Exception will be thrown
- NoSuchBeanDefinitionException: No qualifying bean of type 'com.coursecube.spring.Hai' available: which qualifies as autowire candidate. Dependency annotations: {Autowired(required=true)}

Case 2: What happens when exactly 1 Matching bean found.(Refer Lab23)

- Identified single bean will be Injected Directly without any setter method.

Case 3: What happens when two or more Matching beans found.

- If Any bean name is matching with local variable name then that will be injected with setter method.**(Refer Lab24)**

- If Any bean name is not matching with local variable name then Exception will be thrown.**(Ref.Lab25)**
- org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type 'com.coursecube.spring.Hai' available: expected single matching bean but found 3: myhai1,myhai2,myhai3

Lab22: Files required

1. **Lab22.java**	2. **Hai.java**
3. **Hello.java**	4. **JLCAppConfig.java**

1. Lab22.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab22 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();
}
}
```

2. Hai.java

```
package com.coursecube.spring;

/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai  {
        String msg;

       public void setMsg(String msg) {
                this.msg = msg;
        }

        public String  toString() {
        return msg;
       }
}
```

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Autowired
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}

}
```

Lab23: Files required

1. Lab23.java	**Same as Lab22**
2. Hai.java	**Same as Lab22**
3. Hello.java	**Same as Lab22**
4. JLCAppConfig.java	**Updated in Lab23**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */

@Configuration
public class JLCAppConfig {

        @Bean(name="myhai1")
        public Hai createHai() {
                Hai hai=new Hai();
                hai.setMsg("I am Hai Bean 1");
                return hai;
        }

        @Bean(name="myhello")
        public Hello createHello() {
                return new Hello();
        }

}
```

Lab24: Files required

1. Lab24.java	**Same as Lab22**
2. Hai.java	**Same as Lab22**
3. Hello.java	**Same as Lab22**
4. JLCAppConfig.java	**Updated in Lab24**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */

@Configuration
public class JLCAppConfig {

	@Bean(name="myhai1")
	public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	}

	@Bean(name="myhai2")
	public Hai createHai2() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 2");
		return hai;
	}

	@Bean(name="hai")
	public Hai createHai() {
		Hai hai=new Hai();
		hai.setMsg("I am also Hai Bean");
		return hai;
	}

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab25: Files required

1. Lab25.java	**Same as Lab22**
2. Hai.java	**Same as Lab22**
3. Hello.java	**Same as Lab22**
4. JLCAppConfig.java	**Updated in Lab25**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */

@Configuration
public class JLCAppConfig {

	@Bean(name="myhai1")
	public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	}

	@Bean(name="myhai2")
	public Hai createHai2() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 2");
		return hai;
	}

	@Bean(name="myhai3")
	public Hai createHai() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 3");
		return hai;
	}

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

B) @Autowired(required=false)

Case 1: What happens when 0 Matching Beans found.(Refer Lab26)

- Bean Property remains Un-Injected.

Case 2: What happens when exactly 1 Matching bean found.(Refer Lab27)

- Identified single bean will be Injected Directly without any setter method.

Case 3: What happens when two or more Matching beans found.

- If Any bean name is matching with local variable name then that will be injected with setter method.**(Refer Lab28)**

- If Any bean name is not matching with local variable name then Exception will be thrown.**(Ref.Lab29)**
- org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type 'com.coursecube.spring.Hai' available: expected single matching bean but found 3: myhai1,myhai2,myhai3

Lab26: Files required

1. Lab26.java	**Same as Lab22**
2. Hai.java	**Same as Lab22**
3. Hello.java	**Updated in Lab26**
4. JLCAppConfig.java	**Updated in Lab26**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Autowired(required = false)
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab27: Files required

1. Lab27.java	**Same as Lab22**
2. Hai.java	**Same as Lab22**
3. Hello.java	**Updated in Lab27**
4. JLCAppConfig.java	**Updated in Lab27**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Autowired(required = false)
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
	@Bean(name="myhai1")
	public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	}
	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab28: Files required

5. Lab28.java	**Same Lab22**
6. Hai.java	**Same Lab22**
7. Hello.java	**Updated in Lab28**
8. JLCAppConfig.java	**Updated in Lab28**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
	@Autowired(required = false)
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */

@Configuration
public class JLCAppConfig {

	@Bean(name="myhai1")
	public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	}

	@Bean(name="myhai2")
	public Hai createHai2() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 2");
		return hai;
	}

	@Bean(name="hai")
	public Hai createHai() {
		Hai hai=new Hai();
		hai.setMsg("I am also Hai Bean");
		return hai;
	}

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab29: Files required

1. Lab29.java	**Same Lab22**
2. Hai.java	**Same Lab22**
3. Hello.java	**Updated in Lab29**
4. JLCAppConfig.java	**Updated in Lab29**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Autowired(required = false)
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhai1")
	public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	}
```

```
        @Bean(name="myhai2")
        public Hai createHai2() {
                Hai hai=new Hai();
                hai.setMsg("I am Hai Bean 2");
                return hai;
        }

        @Bean(name="myhai3")
        public Hai createHai() {
                Hai hai=new Hai();
                hai.setMsg("I am Hai Bean 3");
                return hai;
        }

        @Bean(name="myhello")
        public Hello createHello() {
                return new Hello();
        }
}
```

2) ByName autowiring with @Autowired

When you want to detect the beans based on byName autowire process then you need to use @Qualifier Annotation along with @Autowired.

A) @Autowired(required=true) / @Autowired

```
@Autowired(required=true)
@Qualifier("myhai")
```

Case 1: What happens when 0 Matching Beans found.(Refer Lab30)

- Exception will be thrown
- NoSuchBeanDefinitionException: No qualifying bean of type 'com.coursecube.spring.Hai' available: which qualifies as autowire candidate. Dependency annotations: {Autowired(required=true)}

Case 2: What happens when exactly 1 Matching bean found.(Refer Lab31)

- Identified single bean will be Injected Directly without any setter method.

Lab30: Files required

1. Lab30.java	**Same Lab22**
2. Hai.java	**Same Lab22**
3. Hello.java	**Updated in Lab30**
4. JLCAppConfig.java	**Updated in Lab30**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Autowired
	@Qualifier("myhai")
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab31: Files required

1. Lab31.java	**Same Lab22**
2. Hai.java	**Same Lab22**
3. Hello.java	**Updated in Lab31**
4. JLCAppConfig.java	**Updated in Lab31**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
	@Autowired
	@Qualifier("myhai")
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
	 @Bean(name="myhai")
	public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	}
	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

B) @Autowired(required=false)

B) @Autowired(required=false)
@Qualifier("myhai")

Case 1: What happens when 0 Matching Beans found.(Refer Lab32)

- Bean Property remains Un-Injected.

Case 2: What happens when exactly 1 Matching bean found.(Refer Lab33)

- Identified single bean will be Injected Directly without any setter method.

Lab32: Files required

5. Lab32.java	**Same Lab22**
6. Hai.java	**Same Lab22**
7. Hello.java	**Updated in Lab32**
8. JLCAppConfig.java	**Updated in Lab32**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Autowired(required=false)
	@Qualifier("myhai")
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab33: Files required

5. Lab33.java	**Same Lab22**
6. Hai.java	**Same Lab22**
7. Hello.java	**Updated in Lab33**
8. JLCAppConfig.java	**Updated in Lab33**

3. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
	@Autowired(required=false)
	@Qualifier("myhai")
	Hai hai; //1

	public void show() {
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
        @Bean(name="myhai")
        public Hai createHai1() {
                Hai hai=new Hai();
                hai.setMsg("I am Hai Bean 1");
                return hai;
        }
        @Bean(name="myhello")
        public Hello createHello() {
                return new Hello();
        }
}
```

Using @Autowired for Setter Methods

- You can use @Autowired for setter methods, then beans will be injected withSetter methods

@Autowired can be used for setter methods in two ways.

1) @Autowired(required=true) **(Refer Lab34)**

```
@Autowired
public void setAobj(A aobj) {
        this.aobj = aobj;
}
```

2) @Autowired(required=false) **(Refer Lab35)**

```
@Autowired(required=false)
public void setAobj(A aobj) {
        this.aobj = aobj;
}
```

Lab34: Files required

1. Lab34.java	2. A.java
3. B.java	4. Hai.java
5. Hello.java	6. JLCAppConfig.java

1. Lab34.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab34 {
public static void main(String[] args) {
ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();
}
}
```

2. A.java

```
package com.coursecube.spring;

/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class A {
  String msg; //S.I

public void setMsg(String msg) {
        System.out.println("A - setMsg()");
        this.msg=msg;
}
public String  toString() {
        return msg;
}
}
```

3. B.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class B {
String str; //C.I

public B( String str) {
        System.out.println("B -1 arg");
        this.str = str;
}
public String  toString() {
        return str;
}
}
```

4. Hai.java

```
package com.coursecube.spring;

/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai  {
        String msg; //Dependency

        public void setMsg(String msg) {
                this.msg = msg;
        }
        public String  toString() {
        return msg;
       }
}
```

5. Hello.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
```

```
public class Hello {

	@Autowired
	private Hai hai;    //Field Injection

	private A  aobj;   //Setter Injection
	private B  bobj;  //Setter Injection

	@Autowired
	public void setAobj(A aobj) {
		System.out.println("Hello-setAobj()");
		this.aobj = aobj;
	}

	@Autowired
	public void setBobj(B bobj) {
		System.out.println("Hello-setBobj()");
		this.bobj = bobj;
	}
	public void show() {
		System.out.println("Hello-show()");
		System.out.println(hai);
		System.out.println(aobj);
		System.out.println(bobj);
	}
}
```

6. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhai")
	public Hai createHai() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean");
		return hai;
	}
```

```
	@Bean(name="myao")
	public A createA() {
		A ao=new A();
		ao.setMsg("I am Bean - A");
		return ao;
	}

	@Bean(name="mybo")
	public B createB() {
		B bo=new B("I am Bean - B");
		return bo;
	}

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab35: Files required

1. Lab35.java	**Same as Lab34**
2. A.java	**Same as Lab34**
3. B.java	**Same as Lab34**
4. Hai.java	**Same as Lab34**
5. Hello.java	**Updated in Lab35**
6. JLCAppConfig.java	**Updated in Lab35**

5. Hello.java

```
package com.coursecube.spring;
import org.springframework.beans.factory.annotation.Autowired;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
	@Autowired
	private Hai hai;    //Field Injection

	private A  aobj;   //Setter Injection
	private B  bobj;  //Setter Injection

	@Autowired (required=false)
	public void setAobj(A aobj) {
		System.out.println("Hello-setAobj()");
		this.aobj = aobj;
	}
```

```
	@Autowired (required=false)
	public void setBobj(B bobj) {
		System.out.println("Hello-setBobj()");
		this.bobj = bobj;
	}
	public void show() {
		System.out.println("Hello-show()");
		System.out.println(hai);
		System.out.println(aobj);
		System.out.println(bobj);
	}
}
```

6. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhai")
	public Hai createHai() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean");
		return hai;
	}

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Using JSR-250 Annotations

- **Following annotations provided in javax.annotation package**
  1) **@PostConstruct** **init()**
  2) **@PreDestroy** **destroy()**
  3) **@Resource**

Note:

- When you want to use JSR-250 Annotations you must add javaee.jar file to project build path.

1) @PostConstruct:

- You can mark the method with @PostConstruct Annotation.
- Method which is marked with @PostConstruct Annotation
  - will be called by the Spring Container after creating Bean Instance
  - contains the code for initializing bean instance with the required resources.

2) @PreDestroy:

- You can mark the method with @ PreDestroy Annotation.
- Method which is marked with @ PreDestroy Annotation
  - will be called by the Spring Container before destroying Bean Instance
  - contains the code for cleaning resources initialized with bean instance.

3) @Resource:

- You can mark the Bean Property with @Resource Annotation.
- When you use @Resource, then beans will be detected either based on byName or byType process and injects them.
  - When name attribute is specified for @Resource then uses byName autowire process. **(Refer Lab36)**
  - When name attribute is not specified for @Resource then uses byType autowire process. **(Refer Lab37)**

Lab36: Files required

1. Lab36.java	**Same Lab22**
2. Hai.java	**Same Lab22**
3. Hello.java	**Updated in Lab36**
4. JLCAppConfig.java	**Updated in Lab36**

3. Hello.java

```
package com.coursecube.spring;

import javax.annotation.Resource;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
```

```
public class Hello {

	@Resource  //ByType(Default)
	Hai hai;   //Field Injection

	public void show() {
		System.out.println("Hello-show()");
		System.out.println(hai);
	}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	 @Bean(name="myhai")
	public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	}

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```

Lab37: Files required

1. Lab37.java	**Same Lab22**
2. Hai.java	**Same Lab22**
3. Hello.java	**Updated in Lab37**
4. JLCAppConfig.java	**Updated in Lab37**

3. Hello.java

```
package com.coursecube.spring;

import javax.annotation.Resource;
/*
```

```
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Resource(name="myhai2")   //ByName
	 Hai hai;    //Field Injection

	 public void show() {
		System.out.println("Hello-show()");
		System.out.println(hai);
	 }
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhai1")
	 public Hai createHai1() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 1");
		return hai;
	 }

	 @Bean(name="myhai2")
	 public Hai createHai2() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean 2");
		return hai;
	 }

	 @Bean(name="myhello")
	 public Hello createHello() {
		return new Hello();
	 }
}
```

@Inject

- When you use @Inject, then by default, beans will be detected based on byType process and inject them directly without any Stetter methods or Constructor.
- When you want to detect the beans based on byName process then you need to @Qualifier Annotation along with @Inject.
- **Note:** When you want to use @Inject Annotation then you must add javax.inject.jar file to project build path.

Lab38: Files required

1. **Lab38.java**	2. **A.java**
3. **Hai.java**	4. **Hello.java**
5. **JLCAppConfig.java**	

1. Lab38.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab38 {
public static void main(String[] args) {
ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");
Hello hello=(Hello)ctx.getBean("myhello");
hello.show();
}
}
```

2. A.java

```
package com.coursecube.spring;
public class A  {
          String str; //Dependency
          public void setStr(String str) {
                    this.str = str;
          }
         public String  toString() {
          return str;
        }
}
```

3. Hai.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai  {
	String msg; //Dependency

public void setMsg(String msg) {
		this.msg = msg;
	}

public String  toString() {
	return msg;
}
}
```

4. Hello.java

```
package com.coursecube.spring;

import javax.inject.Inject;
import org.springframework.beans.factory.annotation.Qualifier;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

	@Inject
	Hai hai;   //ByType

	@Inject
	@Qualifier("myao")
	A aobj;    //ByName

	public void show() {
		System.out.println("Hello-show()");
		System.out.println(hai);
		System.out.println(aobj);
	}
}
```

5. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhai")
	public Hai createHai() {
		Hai hai=new Hai();
		hai.setMsg("I am Hai Bean");
		return hai;
	}

	@Bean(name="myao")
	public A createA() {
		A ao=new A();
		ao.setStr("I am Bean - A");
		return ao;
	}

	@Bean(name="myhello")
	public Hello createHello() {
		return new Hello();
	}
}
```