SPRING FRAMEWORK AUTOWIRE SCENARIOS

App.java

```
package org.studyeasy;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.studyeasy.interfaces.Car;

public class App {
public static void main(String[] args) {

// Outsourcing Object Creation process
AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

// Outsource the work to Spring.
Car myCar = context.getBean("co", Car.class);
myCar.specs();

// Closing the context also
context.close();

}
}
```

Engine.java

```
package org.studyeasy.cars;

import org.springframework.stereotype.Component;

@Component
public class Engine {
String type = "V8";
}
```

Corolla.java

```
package org.studyeasy.cars;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.studyeasy.interfaces.Car;

@Component("co")
public class Corolla implements Car {

	@Autowired
	Engine engine;

	@Override
	public void specs() {
		System.out.println("Sedan from Toyota wit engine as : " +engine.type);
	}
}
```

Swift.java

```
package org.studyeasy.cars;

import org.springframework.stereotype.Component;
import org.studyeasy.interfaces.Car;

@Component("sw")
public class Swift implements Car{

	@Override
	public void specs() {
		System.out.println("Hatchback from Suzuki");
	}
}
```

AppConfig.java

```
package org.studyeasy;

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

// Telling Spring that it is configuration class and which particular package to scan for
// components

@Configuration
@ComponentScan("org.studyeasy")
public class AppConfig {
}
```

Car.java

```
package org.studyeasy.interfaces;

public interface Car {
void specs();
}
```

Scenario 1:

If we have a parametrized constructor, there will be no default constructor inside this class by Java.

When the Spring will try to create a object using the bean factory, we won't be able to intialize the reference and application will crash. (Engine.java)

In short, the application will crash because there is no way to automatically create a object without a default constructor. (Engine.java).

Adding a Parameterized Constructor to the class Engine

```
public Engine(String type) {
super();
this.type = type;
}
```

You get error

```
May 17, 2024 9:54:37 PM org.springframework.context.support.AbstractApplicationContext refresh
WARNING: Exception encountered during context initialization - cancelling refresh attempt:
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'co': Unsatisfied dependency expressed through
field 'engine': Error creating bean with name 'engine' defined in file [D:\RECYCLE BIN
EXTRAS\ONLY_DOCS\JavaUdemy\Java_SpringFramework_AutoWireScenarios\SPF05-Spring_AutowireContinued\target\classes\org\studyeasy\cars\Engine.
class]: Unsatisfied dependency expressed through constructor parameter 0: No qualifying bean of type 'java.lang.String' available: expected
at least 1 bean which qualifies as autowire candidate. Dependency annotations: {}
Exception in thread "main" org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'co': Unsatisfied
dependency expressed through field 'engine': Error creating bean with name 'engine' defined in file [D:\RECYCLE BIN
EXTRAS\ONLY_DOCS\JavaUdemy\Java_SpringFramework_AutoWireScenarios\SPF05-Spring_AutowireContinued\target\classes\org\studyeasy\cars\Engine.
class]: Unsatisfied dependency expressed through constructor parameter 0: No qualifying bean of type 'java.lang.String' available: expected
at least 1 bean which qualifies as autowire candidate. Dependency annotations: {}
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.resolveFieldValue(AutowiredAnnota
tionBeanPostProcessor.java:787)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotationBeanPos
tProcessor.java:767)
at org.springframework.beans.factory.annotation.InjectionMetadata.inject(InjectionMetadata.java:145)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor.postProcessProperties(AutowiredAnnotationBeanPostProces
sor.java:508)
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBeanFactory.java:1419)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:599)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:522)
at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:326)
at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:234)
at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:324)
at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:200)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:975)
at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:962)
at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:624)
at org.springframework.context.annotation.AnnotationConfigApplicationContext.<init>(AnnotationConfigApplicationContext.java:93)
at org.studyeasy.App.main(App.java:18)
Caused by: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'engine' defined in file [D:\RECYCLE
BIN
EXTRAS\ONLY_DOCS\JavaUdemy\Java_SpringFramework_AutoWireScenarios\SPF05-Spring_AutowireContinued\target\classes\org\studyeasy\cars\Engine.
class]: Unsatisfied dependency expressed through constructor parameter 0: No qualifying bean of type 'java.lang.String' available: expected
at least 1 bean which qualifies as autowire candidate. Dependency annotations: {}
at org.springframework.beans.factory.support.ConstructorResolver.createArgumentArray(ConstructorResolver.java:795)
at org.springframework.beans.factory.support.ConstructorResolver.autowireConstructor(ConstructorResolver.java:237)
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.autowireConstructor(AbstractAutowireCapableBeanFactory.java:
1355)
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBeanInstance(AbstractAutowireCapableBeanFactory.java:1
192)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:562)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:522)
at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:326)
at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:234)
at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:324)
at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:200)
at org.springframework.beans.factory.config.DependencyDescriptor.resolveCandidate(DependencyDescriptor.java:254)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.java:1443)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.java:1353)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.resolveFieldValue(AutowiredAnnota
tionBeanPostProcessor.java:784)
... 15 more
Caused by: org.springframework.beans.factory.NoSuchBeanDefinitionException: No qualifying bean of type 'java.lang.String' available: expected
at least 1 bean which qualifies as autowire candidate. Dependency annotations: {}
at org.springframework.beans.factory.support.DefaultListableBeanFactory.raiseNoMatchingBeanFound(DefaultListableBeanFactory.java:1880)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.java:1406)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.java:1353)
at org.springframework.beans.factory.support.ConstructorResolver.resolveAutowiredArgument(ConstructorResolver.java:904)
at org.springframework.beans.factory.support.ConstructorResolver.createArgumentArray(ConstructorResolver.java:782)
... 28 more
```

The Solution is Get rid of the parameterized constructor and in order to create a constructor we will create the constructor for the property directly inside the bean class (Corolla.java).

In Corolla.java

```
public Corolla(Engine engine) {
engine.type = "V6";
//this.engine = engine; -> This is optional
}
```

We wil get ouput as :- **Sedan from Toyota wit engine as : V6**

Somewhat we are able to overide and get output.

In Corolla.java : Below following is also fine

```
public Corolla(Engine engine) {
engine.type = "V6";
//this.engine = engine;
}
```

```
@Autowired
public Corolla(Engine engine) {
engine.type = "V6";
}
```

```
@Autowired
public Corolla(Engine engine) {
engine.type = "V6";
//this.engine = engine; -> If we keep this line it is also fine
}
```

As it is called or executed as soon as object created . (Engine engine;)

We get Error like below:-

```
Exception in thread "main" java.lang.NullPointerException: Cannot read field "type" because
"this.engine" is null
at org.studyeasy.cars.Corolla.specs(Corolla.java:23)
at org.studyeasy.App.main(App.java:14)
```

Solution

```
@Autowired
public Corolla(Engine engine) {
engine.type = "V6";
this.engine = engine; <- Uncomment this out
}
```

We can autowire the constructor or the reference both - No issue at all.

Scenario 2: Using getter

Commenting out the constructor and using getter. Add getter in Corolla.java.

```
public void setEngine(Engine engine) {
this.engine = engine;
}
```

Add below line in above getter

```
engine.type = "V10";
```

The Output:-

```
Sedan from Toyota wit engine as : V8
```

The reason is the setter work in a different way. When we autowire the reference, a new object with the default constructor will be created. We will se the default value only.

To run the setter, it is essential to use autowire on the setter.

```
@Autowired
public void setEngine(Engine engine) {
engine.type = "V10";
this.engine = engine;
}
```

Output

```
Sedan from Toyota wit engine as : V10
```

```
@Autowired
public void setEngine(Engine engine) {
engine.type = "V10";
//this.engine = engine; <- Comment out this line you app will crash
}
```

You get output like below

```
Exception in thread "main" java.lang.NullPointerException: Cannot read field "type" because "this.engine"
is null
at org.studyeasy.cars.Corolla.specs(Corolla.java:46)
at org.studyeasy.App.main(App.java:14)
```

Qualifier Annotation

Let us say that Engine class is a interface. Make and modify it as interface. Move Engine Class to interfaces package.

```
package org.studyeasy.interfaces;

public interface Engine {
String specs();
}
```

Create a new Class -> **Package name** : org.studyeasy.specs

Class Name : V8 -> It should also implement Engine Interface

Create a new Class -> **Package name** : org.studyeasy.specs

Class Name : V6 -> It should also implement Engine Interface

V6.java

```
package org.studyeasy.specs;

import org.studyeasy.interfaces.Engine;

public class V6 implements Engine{

@Override
public String specs() {
return "V6 Engine";
}
}
```

V8.java

```java
package org.studyeasy.specs;

import org.studyeasy.interfaces.Engine;

public class V8 implements Engine {

@Override
public String specs() {
return "V8 Engine";
}
}
```

These 2 Classes are Components as well. So mark both classes with **@Component**.

In Corolla.java

```java
package org.studyeasy.cars;

import org.springframework.stereotype.Component;
import org.studyeasy.interfaces.Car;
import org.studyeasy.interfaces.Engine;

@Component("co")
public class Corolla implements Car {

@Autowired
Engine engine;

@Override
public void specs() {
System.out.println("Sedan from Toyota wit engine as : " + engine.specs());
}
}
```

Output:- The applictaion will crash.

```
May 17, 2024 11:07:07 PM org.springframework.context.support.AbstractApplicationContext refresh
WARNING: Exception encountered during context initialization - cancelling refresh attempt:
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'co': Unsatisfied dependency expressed through
field 'engine': No qualifying bean of type 'org.studyeasy.interfaces.Engine' available: expected single matching bean but found 2: v6,v8
Exception in thread "main" org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'co': Unsatisfied
dependency expressed through field 'engine': No qualifying bean of type 'org.studyeasy.interfaces.Engine' available: expected single matching
bean but found 2: v6,v8
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.resolveFieldValue(AutowiredAnnota
tionBeanPostProcessor.java:787)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotationBeanPos
tProcessor.java:767)
at org.springframework.beans.factory.annotation.InjectionMetadata.inject(InjectionMetadata.java:145)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor.postProcessProperties(AutowiredAnnotationBeanPostProces
sor.java:508)
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBeanFactory.java:1419)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:599)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:522)
at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:326)
at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:234)
at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:324)
at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:200)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:975)
at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:962)
at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:624)
at org.springframework.context.annotation.AnnotationConfigApplicationContext.<init>(AnnotationConfigApplicationContext.java:93)
at org.studyeasy.App.main(App.java:10)
Caused by: org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type
'org.studyeasy.interfaces.Engine' available: expected single matching bean but found 2: v6,v8
at org.springframework.beans.factory.config.DependencyDescriptor.resolveNotUnique(DependencyDescriptor.java:218)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.java:1420)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.java:1353)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.resolveFieldValue(AutowiredAnnota
tionBeanPostProcessor.java:784)
... 15 more
```

Give Component name to both V6 and V8 classes.

```
@Component("V6Engine")
@Component("V8Engine")
```

The System will crash again.

```
May 17, 2024 11:12:50 PM org.springframework.context.support.AbstractApplicationContext refresh
WARNING: Exception encountered during context initialization - cancelling refresh attempt:
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'co': Unsatisfied
dependency expressed through field 'v6': No qualifying bean of type 'org.studyeasy.interfaces.Engine' available: expected
single matching bean but found 2: V6Engine,V8Engine
Exception in thread "main" org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with
name 'co': Unsatisfied dependency expressed through field 'v6': No qualifying bean of type
'org.studyeasy.interfaces.Engine' available: expected single matching bean but found 2: V6Engine,V8Engine
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.resolveField
Value(AutowiredAnnotationBeanPostProcessor.java:787)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(Autow
iredAnnotationBeanPostProcessor.java:767)
at org.springframework.beans.factory.annotation.InjectionMetadata.inject(InjectionMetadata.java:145)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor.postProcessProperties(AutowiredAnn
otationBeanPostProcessor.java:508)
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBea
nFactory.java:1419)
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBea
nFactory.java:599)
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanF
actory.java:522)
at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:326)
at
org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.jav
a:234)
at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:324)
at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:200)
at
org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFac
tory.java:975)
at
org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationCo
ntext.java:962)
at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:624)
at
org.springframework.context.annotation.AnnotationConfigApplicationContext.<init>(AnnotationConfigApplicationContext.
java:93)
at org.studyeasy.App.main(App.java:10)
Caused by: org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean
of type 'org.studyeasy.interfaces.Engine' available: expected single matching bean but found 2:
V6Engine,V8Engine
at org.springframework.beans.factory.config.DependencyDescriptor.resolveNotUnique(DependencyDescriptor.java:218)
at
org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.
java:1420)
at
org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.ja
va:1353)
at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.resolveField
Value(AutowiredAnnotationBeanPostProcessor.java:784)
... 15 more
```

In Corolla.java

```
// Case 3
@Autowired
Engine v6Engine;


@Override
public void specs() {
System.out.println("Sedan from Toyota wit engine as : " + v6Engine.specs());
}
```

You get output like below. -> `Sedan from Toyota wit engine as : V6 Engine`

```
// Case 4
@Autowired
Engine engine;
```

```
Add the annotation to make it better code -> @Qualifier("V6Engine")
```

Spring Beans/Spring Beans Annotation

Get rid of the @Component annotations from below classes.
- ✓ Corolla.java
- ✓ Swift.java
- ✓ V6.java
- ✓ V8.java

There were lot of annotations and imports in different files which can make our life or app complicated.

In order to make it simple, it is better idea to make use of concept of Spring Bean.

In App.Config

```
package org.studyeasy;

import org.springframework.context.annotation.Bean;
import org.studyeasy.cars.Corolla;
import org.studyeasy.cars.Swift;
import org.studyeasy.specs.V6;
import org.studyeasy.specs.V8;

public class AppConfig {
@Bean("corolla")
public Corolla corolla(){
return new Corolla();
}
@Bean("swift")
public Swift swift(){
return new Swift();
}
@Bean("V6Engine")
public V6 v6(){
return new V6();
}
@Bean("V8Engine")
public V8 v8(){
return new V8();
}
}
```

These Methods are nothing but replacement of your components.
All the beans in single file will make you easy. Even Constructor Injection will be easy.

Constructor Injection

- ✓ Delete V8 Class -> V8.java. Create a property in V6.java like below.
  It will be using Default and Parametrized Constructor.

- ✓ Rename the file V6.java to Enginetype.java.

- ✓ Modify the AppConfig.java -> Remove V8() method and few other changes. The final code is below.

Final Code.

```
package org.studyeasy.specs;

import org.studyeasy.interfaces.Engine;

public class EngineType implements Engine{

String type;
public EngineType() {
type = "Unknown";
}

public EngineType(String type) {
super();
this.type = type;
}

@Override
public String specs() {
return type;
}
}
```

AppConfig.java

```
package org.studyeasy;

import org.springframework.context.annotation.Bean;
import org.studyeasy.cars.Corolla;
import org.studyeasy.cars.Swift;
import org.studyeasy.specs.EngineType;

public class AppConfig {
@Bean("co")
public Corolla corolla(){
return new Corolla();
}
@Bean("swift")
public Swift swift(){
return new Swift();
}
@Bean("UnknownEngine")
public EngineType type(){
return new EngineType();
}
}
```

In Corolla.java -> Change

```
@Autowired
@Qualifier("V6Engine")
Engine engine;
```

To

```
@Autowired
@Qualifier("UnknownEngine")
Engine engine;
```

We get Output as -> `Sedan from Toyota wit engine as : Unknown`

Override the constructor by adding below code.

```
@Bean("V6Engine")
public EngineType v6type(){
return new EngineType("V6 Engine");
}
```

In Corolla.java -> If we do not want Unknown Engine and we want V6 engine.

```
@Autowired
@Qualifier("UnknownEngine")
Engine engine;
```

To

```
@Autowired
@Qualifier("V6Engine")
Engine engine;
```

We get Output as -> `Sedan from Toyota wit engine as : V6 Engine`

The final code of App.config

```
package org.studyeasy;

import org.springframework.context.annotation.Bean;
import org.studyeasy.cars.Corolla;
import org.studyeasy.cars.Swift;
import org.studyeasy.specs.EngineType;



public class AppConfig {

     @Bean("co")
     public Corolla corolla(){
          return new Corolla();
     }

     @Bean("swift")
     public Swift swift(){
          return new Swift();
     }

     @Bean("UnknownEngine")
     public EngineType type(){
          return new EngineType();
     }

     @Bean("V6Engine")
     public EngineType v6type(){
          return new EngineType("V6 Engine");
     }

}
```