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)
```