Dependency Injection

Using Annotation & AutoWiring

Spring Container Configuration

- There are 3 ways to configure spring container
  1. FULL XML Config
  2. XML Component Scan
  3. Java Configuration Class (No XML)

Development Process

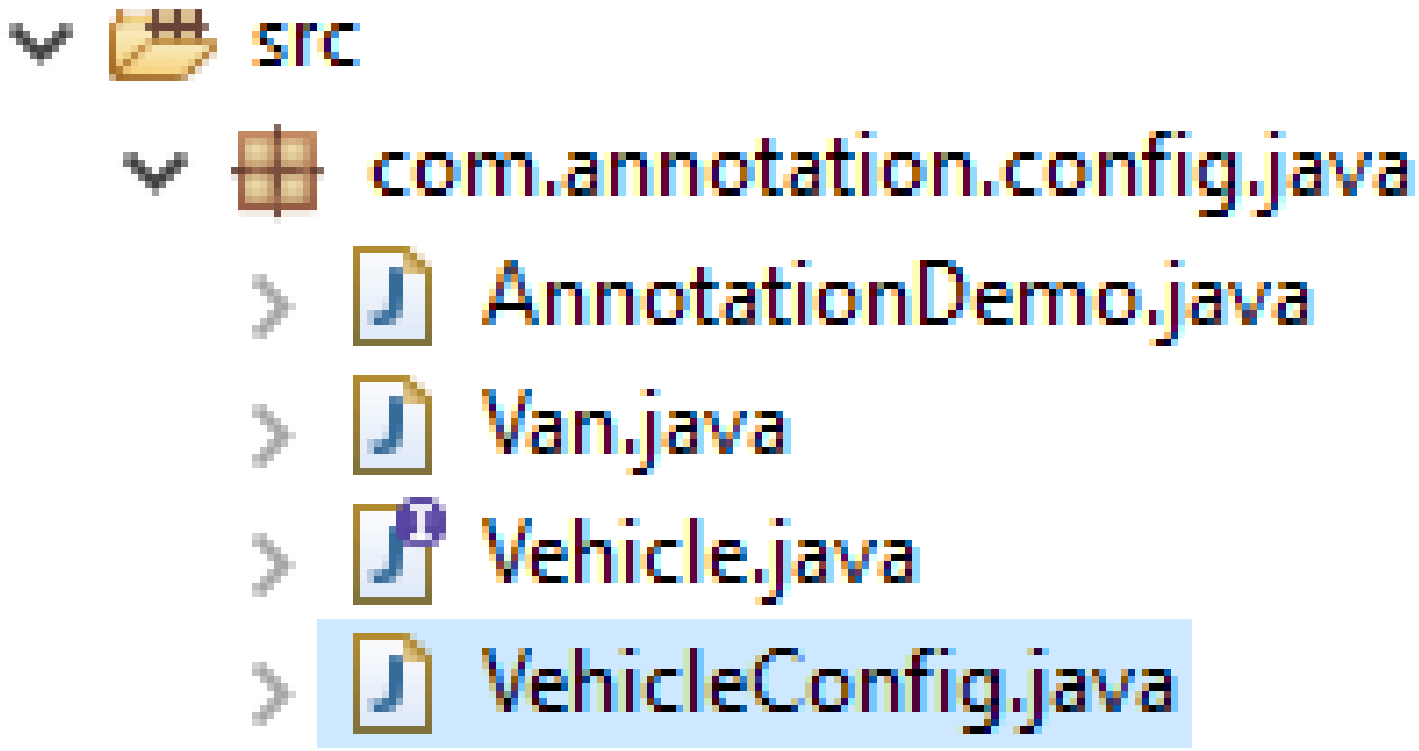
1. Create a Java class and annotate with @Configuration
2. Add component scanning support: @ComponentScan(Optional)
3. Read Spring Java Configuration class
4. Retrieve bean from Spring Container

Development Process

1. Create a simple configuration class
2. Create a Vehicle interface
3. Create Van (bean) class implements Vehicle
4. Create main class
   a) Load java config file using AnnotationConfigApplicationContext
   b) Retrieve bean object from Spring Container
   c) Read & display the info from bean Object

Note: No XML is used for configuration

0. Create a package



1. Create a simple configuration class

VehicleConfig.java

```java
package com.annotation.config.java;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan("com.annotation.config.java")
public class VehicleConfig {

}
```

2. Create a Vehicle interface

Vehicle.java

```java
package com.annotation.config.java;

public interface Vehicle {
    public String getVehicle();
}

```

3. Create Van (bean) class implements Vehicle

Van.java

```java
package com.annotation.config.java;
import org.springframework.stereotype.Component;

@Component
public class Van implements Vehicle {

    public String getVehicle() {
        return "Hi,im using a Van - from annotation";
    }
}
```

4. Create main class

AnnotationDemo.java

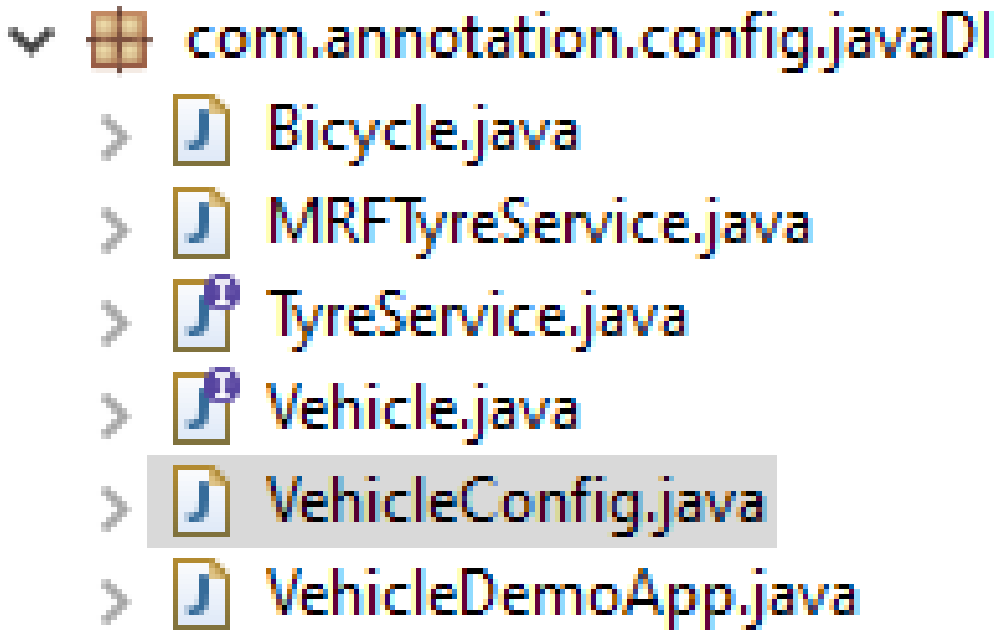
```java
package com.annotation.config.java;
import org.springframework.context.annotation.AnnotationConfigApplicatio

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

        AnnotationConfigApplicationContext context = new
                AnnotationConfigApplicationContext(VehicleConfig.class);

        Vehicle theVehicle = context.getBean("van", Vehicle.class);
        System.out.println(theVehicle.getVehicle());
        context.close();
    }
}
```

DI using java config

Using @Bean with @Componentscan

Development Steps

- Just remove the @ComponentScan in the config file
- Add methods to create and return bean objects annotated with @Bean
- Package Structure



1. Configuration File

VehicleConfig.java

```java
package com.annotation.config.javaDI;
import org.springframework.context.annotation.Bean;

@Configuration
//@ComponentScan("com.annotation.config.java")

public class VehicleConfig {

    @Bean
    public MRFTyreService mrfTyreService() {
        return new MRFTyreService();
    }

    @Bean
    public Bicycle bicycle() {
        return new Bicycle();
    }
}
```

2. Main class

VehicleDemoApp.java

```java
package com.annotation.config.javaDI;
import org.springframework.context.annotation.AnnotationConfigApplicatio

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

        AnnotationConfigApplicationContext context = new
                AnnotationConfigApplicationContext(VehicleConfig.class);

        Vehicle theVehicle = context.getBean("bicycle", Vehicle.class);
        System.out.println(theVehicle.getVehicle());
        System.out.println(theVehicle.getVehicleTyre());

        context.close();
    }
}
```

Vehicle & TyreService interfaces

Vehicle.java

```java
package com.annotation.config.javaDI;

public interface Vehicle {
    public String getVehicle();
    public String getVehicleTyre();
}
```

TyreService.java

```java
package com.annotation.config.javaDI;

public interface TyreService {
    public String getTyre();
}
```

TyreService Class

MRFTyreService.java

```java
package com.annotation.config.javaDI;
import org.springframework.stereotype.Component;

@Component
public class MRFTyreService implements TyreService {

    public String getTyre() {
        return "In my vehicle im using MRF Tyre";
    }
}
```

Bicycle Class

Bicycle.java

```java
1  package com.annotation.config.javaDI;
2  import org.springframework.beans.factory.anno
4
5  @Component
6  public class Bicycle implements Vehicle{
7
8      @Autowired
9      public TyreService tyreService;
10
11     public String getVehicle() {
12         return "Hi, Im using a Bicycle !!!";
13     }
14     public String getVehicleTyre() {
15         return tyreService.getTyre();
16     }
17 }
```

Java Config + properties file

Using @PropertySource and @value

Development Process

- Load properties file in the java config file

```
@PropertySource("classpath: filename")
```

- Read the property values using @Value in Bean Class

```
@Value("${propertykey}")
private String email;
```

Busterminal properties file

- Properties file will be created under src folder

busterminal.properties

```
source = Mumbai
destination = Delhi

```

Load property file

VehicleConfig.java

```
2  import org.springframework.context.annotation.Bean;
5
6  @Configuration
7  @PropertySource("classpath:busterminal.properties")
8  public class VehicleConfig {
9
10     @Bean
11     public Van van() {
12         return new Van();
13     }
14 }
```

Read the property values

Van.java

```java
1  package com.annotation.config.javaprop;
2  import org.springframework.beans.factory.annotation.Value;
4
5  @Component
6  public class Van implements Vehicle {
7
8      @Value("${source}")
9      private String source;
10
11     @Value("${destination}")
12     private String destination;
13
14     public String getVehicle() {
15         return "Hi,im travelling from "+source+" to "+destination;
16     }
17 }
```

Thank You