**JavaBasedSpringConfiguration**

1. In java Based configuration we use a .java file (i.e. ConfigurationClass.java) instead of spring.xml file. And in this file we use @Configuration which makes this java file as like spring.xml file.

**package** com.doj.spring;

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

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

@Configuration

/\*

\* @Configuration annotation is used to define java based configuration this works like spring.xml

\*/

@ComponentScan(basePackages={"com.doj.spring"})

//@ComponentScan by default it search component in current package but we can also define another package also like below

//@ComponentScan(basePackages={"com.org"})//can also define more than one package here because this String array type

**publicclass** SpringConfiguration {

/\*@Bean(name="car")

public Car getCar(){

return new Car();

}\*/

//<bean id="car" class="com.doj.spring.Car"/>

}

1. We also use @ComponentScan annotation in Java Based Configuration file (i.e. ConfigurationClass.java) to scan the @Component annotation. Because @Component will create bean of that class.
2. @ComponentScan by default it search component in current package but we can also define another package also like below

@ComponentScan(basePackages={"com.org"}) can also define more than one package here because this String array type like @ComponentScan(basePackages={"com.org",”com.org1”})

1. We have to use AnnotationConfigApplicationContext class to load Java Based Configuration class in main method which is given at the end of this file.

AbstractApplicationContext context = new AnnotationConfigApplicationContext(SpringConfiguration.class);

1. Other java classes if we want to create bean of those classes then we have to use @Component in those classes.

**Engine.java class**

**package** com.doj.spring;

**import** org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id="engine" class="com.doj.spring.Engine"/>

\*/

@Component

**Public class** Engine {

**final** String engineName="FORD";

**public** String getEngineName() {

**return** engineName;

}

}

1. **Body.java class**

**package** com.org;

**import** org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id="body" class="com.org.Body"/>

\*/

@Component

**Public class** Body {

String bodyName="FORD";

**public** String getBodyName() {

**return** bodyName;

}

}

1. **Main class CarDemo.java**

**package** com.doj.spring;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

**import** org.springframework.context.support.AbstractApplicationContext;

**publicclass** CarDemo {

**publicstaticvoid** main(String[] args) {

//ClassPathXmlApplicationContext is using for XML based configuration

//AbstractApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");

//AnnotationConfigApplicationContext is using for JAVA based configuration

AbstractApplicationContext context = **new** AnnotationConfigApplicationContext(SpringConfiguration.**class**);

Car car = (Car) context.getBean("car");

car.driveCar();

context.close();

}

}

1. **Tyre.java class**

**package** com.doj.spring;

**import** org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id="Tyre" class="com.doj.spring.Tyre"/>

\*/

@Component

**publicclass** Tyre {

String tyreName="MRF";

**public** String getTyreName() {

**return**tyreName;

}

}

1. **Car.java**

**package com.doj.spring;**

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

**import org.springframework.stereotype.Component;**

**/\***

**\* @Component annotation is used to define bean.**

**\* It is like <bean id="car" class="com.doj.spring.Car"/>**

**\*/**

**@Component**

**public class Car {**

**@Autowired(required=false)**

**Engine engine;**

**@Autowired**

**Tyre tyre;**

**@Autowired**

**Body body;**

**public Car(Engine engine, Tyre tyre, Body body) {**

**super();**

**this.engine = engine;**

**this.tyre = tyre;**

**this.body = body;**

**}**

**public Car() {**

**super();**

**}**

**public void driveCar(){**

**System.out.println("You are driving a car which has a engine "+this.engine.engineName**

**+" has a tyre "+this.tyre.getTyreName()+" has a body "+this.body.getBodyType());**

**}**

**}**