Example on @service:

// Content: @Service annotation is used with classes that provide some business functionalities.It can be applied only to class. The spring context will autodelete these class

when annotation-based configuration and classpath scanning is used.

// Annotation based configuration is used here.

// pom.xml dependency to add:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.6.RELEASE</version>

</dependency>

//Create a service class:

package springCoreTest;

import org.springframework.stereotype.Service;

@Service("ms")

public class MathService {

public int add(int x, int y) {

return x + y;

}

public int subtract(int x, int y) {

return x - y;

}

}

//Create a Main class:

package springCoreTest;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class SpringMainClass {

public static void main(String[] args) {

AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext();

context.scan("springCoreTest");

context.refresh();

MathService ms = context.getBean(MathService.class);

int add = ms.add(9, 5);

System.out.println("Addition = " + add);

int subtract = ms.subtract(10, 2);

System.out.println("Subtraction = " + subtract);

context.close();

}

}

output:

Addition = 14

Subtraction = 8

Question 4:

Example on @Configuration:

//Content: This annotation indicates that a class declares one or more @Bean methods and may be processed by the Spring container to generate bean definitions

and service requests for those beans at runtime.

// pom.xml dependency to add:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.9</version>

</dependency>

//Create a class to config:

package com.SpringAnno.ConfigAnno;

public class Audi {

public void config() {

System.out.println("A4, Black Colour, 73L fuel capacity");

}

}

//Create a Main class:

package com.SpringAnno.ConfigAnno;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App

{

public static void main( String[] args )

{

ApplicationContext factory= new AnnotationConfigApplicationContext(AudiCar.class);

Audi A4= factory.getBean(Audi.class);

A4.config();

}

}

// Create a Config class:

package com.SpringAnno.ConfigAnno;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AudiCar {

@Bean

public Audi getCar() {

return new Audi();

}

}

Output:

A4, Black Colour, 73L fuel capacity

Example on @service:

// Content: @Service annotation is used with classes that provide some business functionalities.It can be applied only to class. The spring context will autodelete these class

when annotation-based configuration and classpath scanning is used.

// Annotation based configuration is used here.

// pom.xml dependency to add:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.6.RELEASE</version>

</dependency>

//Create a service class:

package springCoreTest;

import org.springframework.stereotype.Service;

@Service("ms")

public class MathService {

public int add(int x, int y) {

return x + y;

}

public int subtract(int x, int y) {

return x - y;

}

}

//Create a Main class:

package springCoreTest;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class SpringMainClass {

public static void main(String[] args) {

AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext();

context.scan("springCoreTest");

context.refresh();

MathService ms = context.getBean(MathService.class);

int add = ms.add(9, 5);

System.out.println("Addition = " + add);

int subtract = ms.subtract(10, 2);

System.out.println("Subtraction = " + subtract);

context.close();

}

}

output:

Addition = 14

Subtraction = 8

Question 4:

Example on @Configuration:

//Content: This annotation indicates that a class declares one or more @Bean methods and may be processed by the Spring container to generate bean definitions

and service requests for those beans at runtime.

// pom.xml dependency to add:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.9</version>

</dependency>

//Create a class to config:

package com.SpringAnno.ConfigAnno;

public class Audi {

public void config() {

System.out.println("A4, Black Colour, 73L fuel capacity");

}

}

//Create a Main class:

package com.SpringAnno.ConfigAnno;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App

{

public static void main( String[] args )

{

ApplicationContext factory= new AnnotationConfigApplicationContext(AudiCar.class);

Audi A4= factory.getBean(Audi.class);

A4.config();

}

}

// Create a Config class:

package com.SpringAnno.ConfigAnno;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AudiCar {

@Bean

public Audi getCar() {

return new Audi();

}

}

Output:

A4, Black Colour, 73L fuel capacity