1、@Configuration标注在类上，相当于把该类作为spring的xml配置文件中的<beans>，作用为：配置spring容器(应用上下文)

package com.test.spring.support.configuration;

@Configuration

public class TestConfiguration {

public TestConfiguration(){

System.out.println("spring容器启动初始化。。。");

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

相当于：

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context" xmlns:jdbc="http://www.springframework.org/schema/jdbc"

xmlns:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:util="http://www.springframework.org/schema/util" xmlns:task="http://www.springframework.org/schema/task" xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd

http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.0.xsd

http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-4.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd

http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd

http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-4.0.xsd" default-lazy-init="false">

</beans>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15

主方法进行测试：

package com.test.spring.support.configuration;

public class TestMain {

public static void main(String[] args) {

//@Configuration注解的spring容器加载方式，用AnnotationConfigApplicationContext替换ClassPathXmlApplicationContext

ApplicationContext context = new AnnotationConfigApplicationContext(TestConfiguration.class);

//如果加载spring-context.xml文件：

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

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12

从运行主方法结果可以看出，spring容器已经启动了：

八月 11, 2016 12:04:11 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext prepareRefresh

信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@203e25d3: startup date [Thu Aug 11 12:04:11 CST 2016]; root of context hierarchy

spring容器启动初始化。。。

* 1
* 2
* 3

2、@Bean标注在方法上(返回某个实例的方法)，等价于spring的xml配置文件中的<bean>，作用为：注册bean对象

bean类：

package com.test.spring.support.configuration;

public class TestBean {

public void sayHello(){

System.out.println("TestBean sayHello...");

}

public String toString(){

return "username:"+this.username+",url:"+this.url+",password:"+this.password;

}

public void start(){

System.out.println("TestBean 初始化。。。");

}

public void cleanUp(){

System.out.println("TestBean 销毁。。。");

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20

配置类：

package com.test.spring.support.configuration;

@Configuration

public class TestConfiguration {

public TestConfiguration(){

System.out.println("spring容器启动初始化。。。");

}

//@Bean注解注册bean,同时可以指定初始化和销毁方法

//@Bean(name="testNean",initMethod="start",destroyMethod="cleanUp")

@Bean

@Scope("prototype")

public TestBean testBean() {

return new TestBean();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

主方法测试类：

package com.test.spring.support.configuration;

public class TestMain {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(TestConfiguration.class);

//获取bean

TestBean tb = context.getBean("testBean");

tb.sayHello();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

注：   
(1)、@Bean注解在返回实例的方法上，如果未通过@Bean指定bean的名称，则默认与标注的方法名相同；   
  
(2)、@Bean注解默认作用域为单例singleton作用域，可通过@Scope(“prototype”)设置为原型作用域；   
  
(3)、既然@Bean的作用是注册bean对象，那么完全可以使用@Component、@Controller、@Service、@Ripository等注解注册bean，当然需要配置@ComponentScan注解进行自动扫描。

bean类：

package com.test.spring.support.configuration;

//添加注册bean的注解

@Component

public class TestBean {

public void sayHello(){

System.out.println("TestBean sayHello...");

}

public String toString(){

return "username:"+this.username+",url:"+this.url+",password:"+this.password;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

配置类：

@Configuration

//添加自动扫描注解，basePackages为TestBean包路径

@ComponentScan(basePackages = "com.test.spring.support.configuration")

public class TestConfiguration {

public TestConfiguration(){

System.out.println("spring容器启动初始化。。。");

}

//取消@Bean注解注册bean的方式

//@Bean

//@Scope("prototype")

//public TestBean testBean() {

// return new TestBean();

//}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

主方法测试获取bean对象：

public class TestMain {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(TestConfiguration.class);

//获取bean

TestBean tb = context.getBean("testBean");

tb.sayHello();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

sayHello()方法都被正常调用。