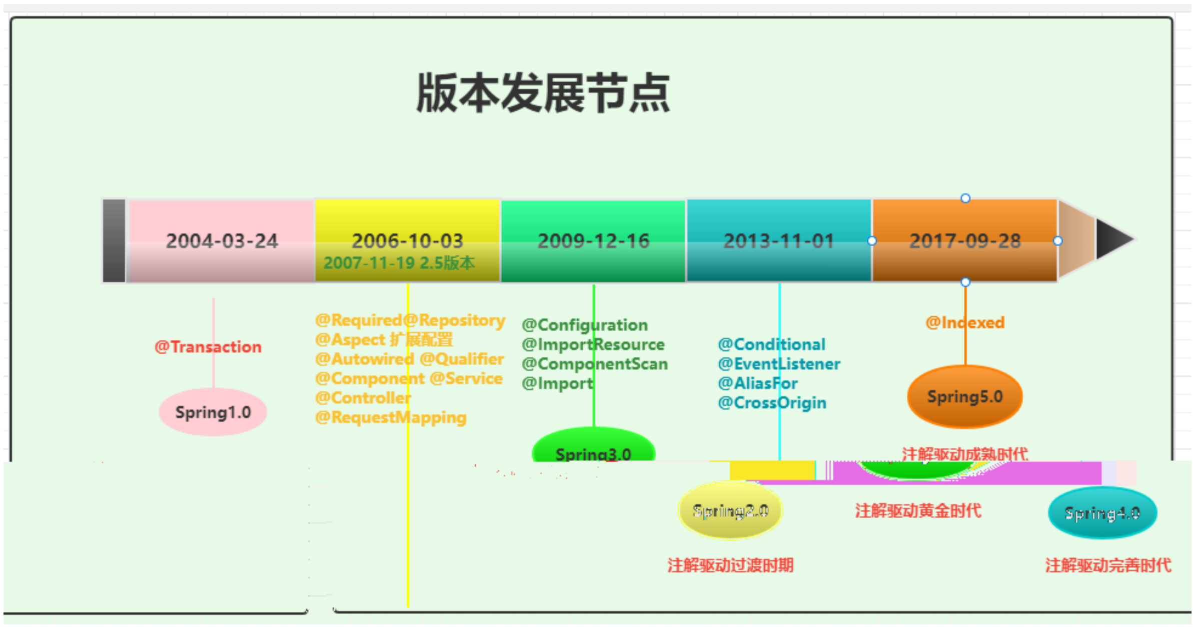
重新认识SpringBoot

1.Spring注解编程的发展过程

SpringBoot Spring
SpringBoot



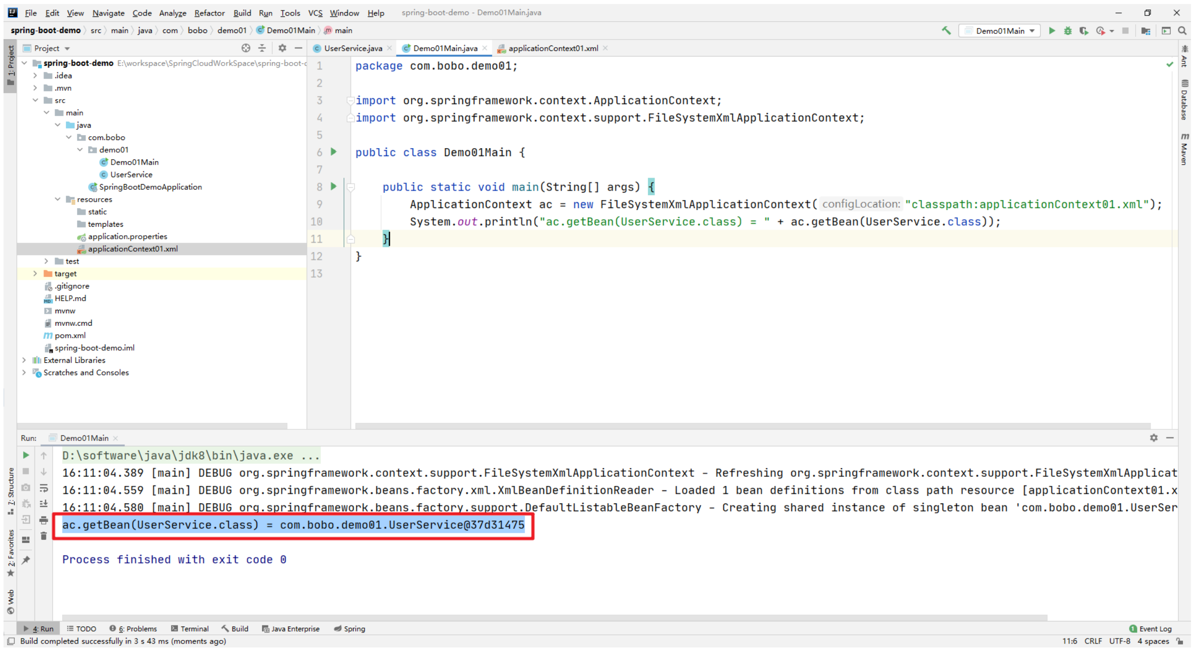
1.1 Spring 1.x

2004 3 24 Spring1.0 IoC AOP XML

Spring1.x XML xml `<bean>` IoC Bean

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">


    <bean class="com.bobo.demo01.UserService" />
</beans>
```

```
public static void main(String[] args) {
    ApplicationContext ac = new
FileSystemXmlApplicationContext("classpath:applicationContext01.xml");
    System.out.println("ac.getBean(UserService.class) = " +
ac.getBean(UserService.class));
}
```



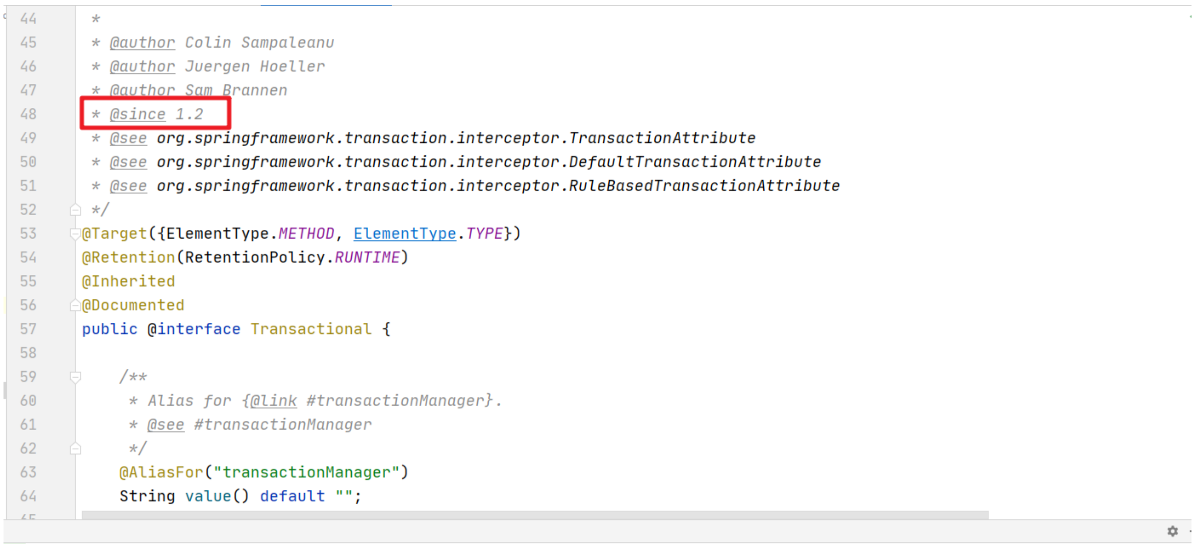
```java
package com.bobo.demo01;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.FileSystemXmlApplicationContext;

public class Demo01Main {

    public static void main(String[] args) {
        ApplicationContext ac = new FileSystemXmlApplicationContext( configLocation: "classpath:applicationContext01.xml");
        System.out.println("ac.getBean(UserService.class) = " + ac.getBean(UserService.class));
    }
}
```

```
D:\software\java\jdk8\bin\java.exe ...
16:11:04.389 [main] DEBUG org.springframework.context.support.FileSystemXmlApplicationContext - Refreshing org.springframework.context.support.FileSystemXmlApplicat
16:11:04.559 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loaded 1 bean definitions from class path resource [applicationContext01.x
16:11:04.580 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'com.bobo.demo01.UserSer
ac.getBean(UserService.class) = com.bobo.demo01.UserService@37d31475

Process finished with exit code 0
```

Build completed successfully in 3 s 43 ms (moments ago)

Spring1.2 @Transaction (org.springframework.transaction.annotation)

.



```java
 *
 * @author Colin Sampaleanu
 * @author Juergen Hoeller
 * @author Sam Brannen
 * @since 1.2
 * @see org.springframework.transaction.interceptor.TransactionAttribute
 * @see org.springframework.transaction.interceptor.DefaultTransactionAttribute
 * @see org.springframework.transaction.interceptor.RuleBasedTransactionAttribute
 */
@Target({ElementType.METHOD, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Inherited
@Documented
public @interface Transactional {

    /**
     * Alias for {@link #transactionManager}.
     * @see #transactionManager
     */
    @AliasFor("transactionManager")
    String value() default "";
```

1.2 Spring 2.x

2006 10 3 Spring2.0 2.x

Spring 2.5之前

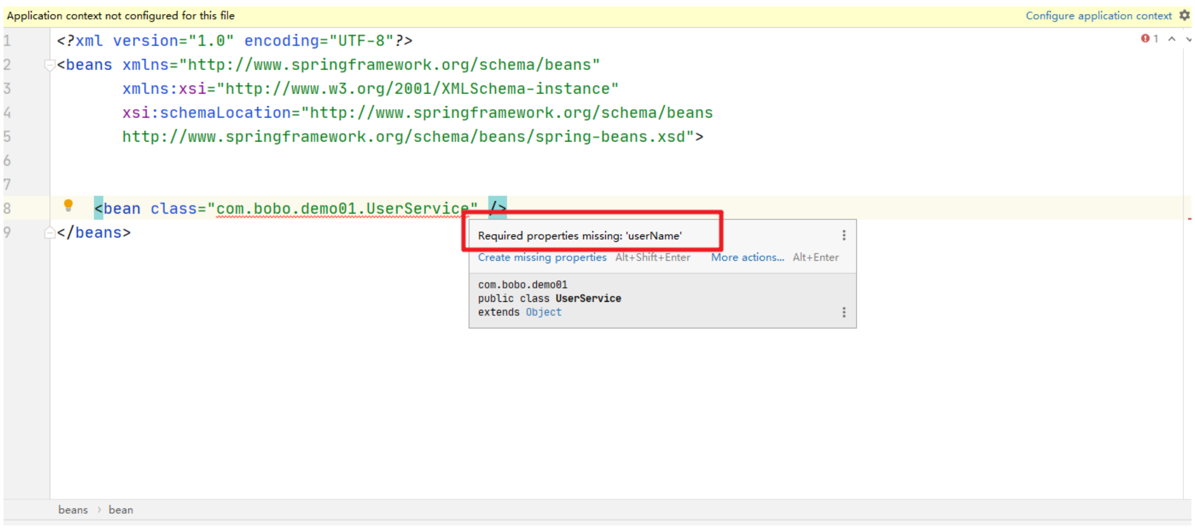
2.5 `@Required` `@Repository` `@Aspect`, XML `<dubbo>`

@Required

java set set xml

```java
public class UserService {


    private String userName;

    public String getUserName() {
        return userName;
    }

    @Required
    public void setUserName(String userName) {
        this.userName = userName;
    }
}
```

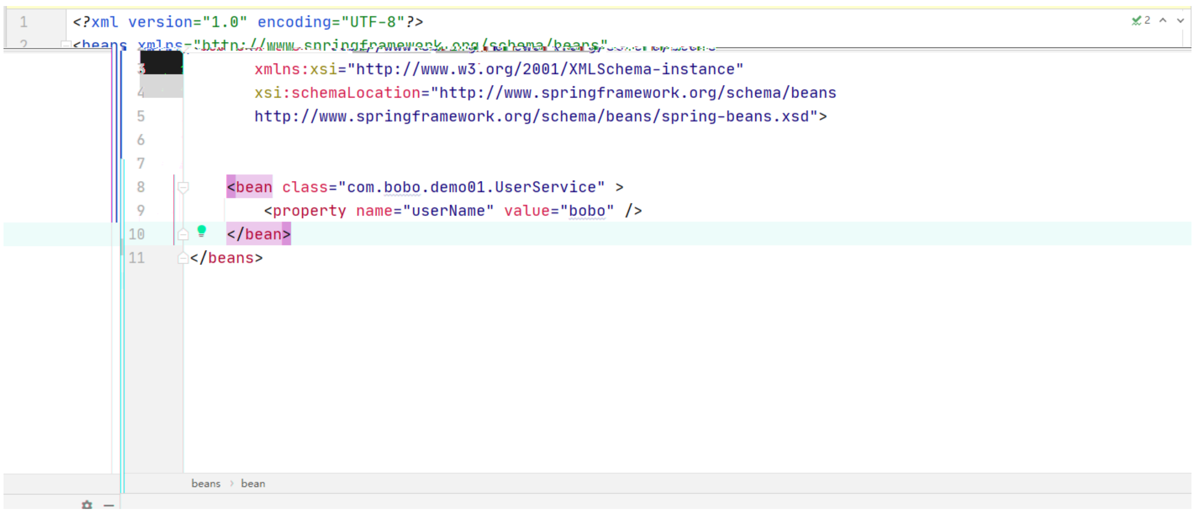
xml



Application context not configured for this file
Configure application context

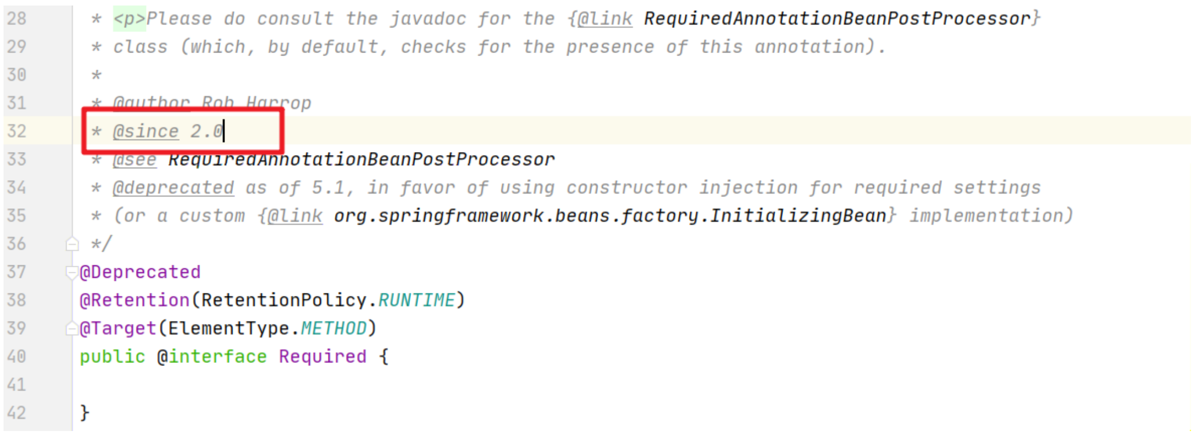
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">


    <bean class="com.bobo.demo01.UserService" />
</beans>
```

```
Required properties missing: 'userName'
Create missing properties  Alt+Shift+Enter    More actions...  Alt+Enter

com.bobo.demo01
public class UserService
extends Object
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">


    <bean class="com.bobo.demo01.UserService" >
        <property name="userName" value="bobo" />
    </bean>
</beans>
```

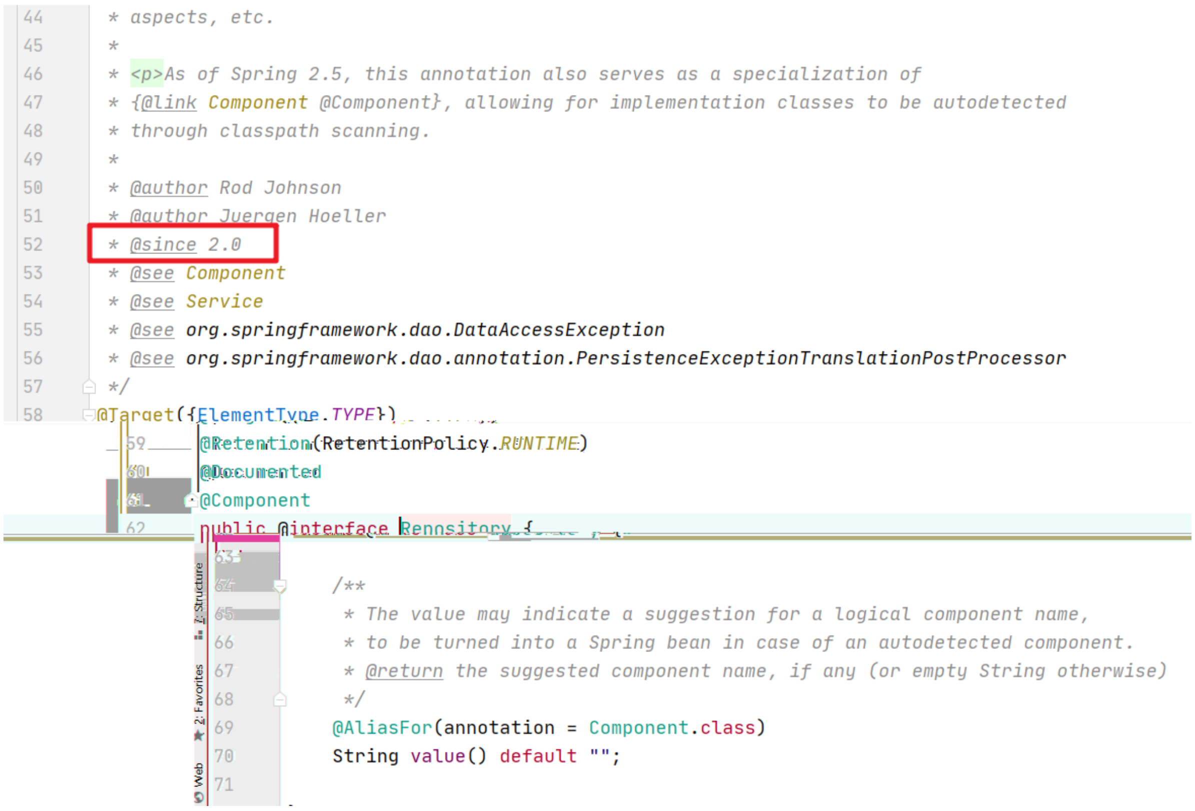
`@Required` 2.0



```java
 * <p>Please do consult the javadoc for the {@link RequiredAnnotationBeanPostProcessor}
 * class (which, by default, checks for the presence of this annotation).
 *
 * @author Rob Harrop
 * @since 2.0
 * @see RequiredAnnotationBeanPostProcessor
 * @deprecated as of 5.1, in favor of using constructor injection for required settings
 * (or a custom {@link org.springframework.beans.factory.InitializingBean} implementation)
 */
@Deprecated
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Required {

}
```

@Repository

@Repository Bean. Spring2.0



```
 * aspects, etc.
 *
 * <p>As of Spring 2.5, this annotation also serves as a specialization of
 * {@link Component @Component}, allowing for implementation classes to be autodetected
 * through classpath scanning.
 *
 * @author Rod Johnson
 * @author Juergen Hoeller
 * @since 2.0
 * @see Component
 * @see Service
 * @see org.springframework.dao.DataAccessException
 * @see org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor
 */
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Component
public @interface Repository {

	/**
	 * The value may indicate a suggestion for a logical component name,
	 * to be turned into a Spring bean in case of an autodetected component.
	 * @return the suggested component name, if any (or empty String otherwise)
	 */
	@AliasFor(annotation = Component.class)
	String value() default "";
```

@Aspect

@Aspect AOP

Spring2.5 之后

2007 11 19 Spring 2.5

注解	说明
@Autowired	
@Qualifier	@Autowired
@Component	
@Service	
@Controller	
@RequestMapping	

xml bean

Bean

```xml
<?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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="com.bobo" />

</beans>
```

```java
@Repository
public class UserDao {

    public void query(){
        System.out.println("dao query ..." );
    }
}
```

```java
@Service
public class UserService {

    @Autowired
    private UserDao dao;

    public void query(){
        dao.query();
    }
}
```

```java
@Controller
public class UserController {

    @Autowired
    private UserService service;

    public void query(){
        service.query();
    }
}
```

```
public class Demo02Main {
    public static void main(String[] args) {
        ApplicationContext ac = new
ClassPathXmlApplicationContext("applicationContext02.xml");
        UserController acBean = ac.getBean(UserController.class);
        acBean.query();
    }
}
```

Spring 2.5 XML

1.3 Spring 3.x

2009 12 16 Spring3.0
Java5 `@Configuration` xml `@ImportResource`
Java XML

```
/**
 * @Configuration 标注的Java类 相当于 application.xml 配置文件
 */
@Configuration
public class JavaConfig {

    /**
     * @Bean 注解 标注的方法就相当于 <bean></bean> 标签
              也是 Spring3.0 提供的注解
     * @return
     */
    @Bean
    public UserService userService(){
        return new UserService();
    }
}
```

Spring3.1 XML `component-scan`
3.1 XML 3.1 `@ComponentScan`
`component-scan` Spring

@ComponentScan

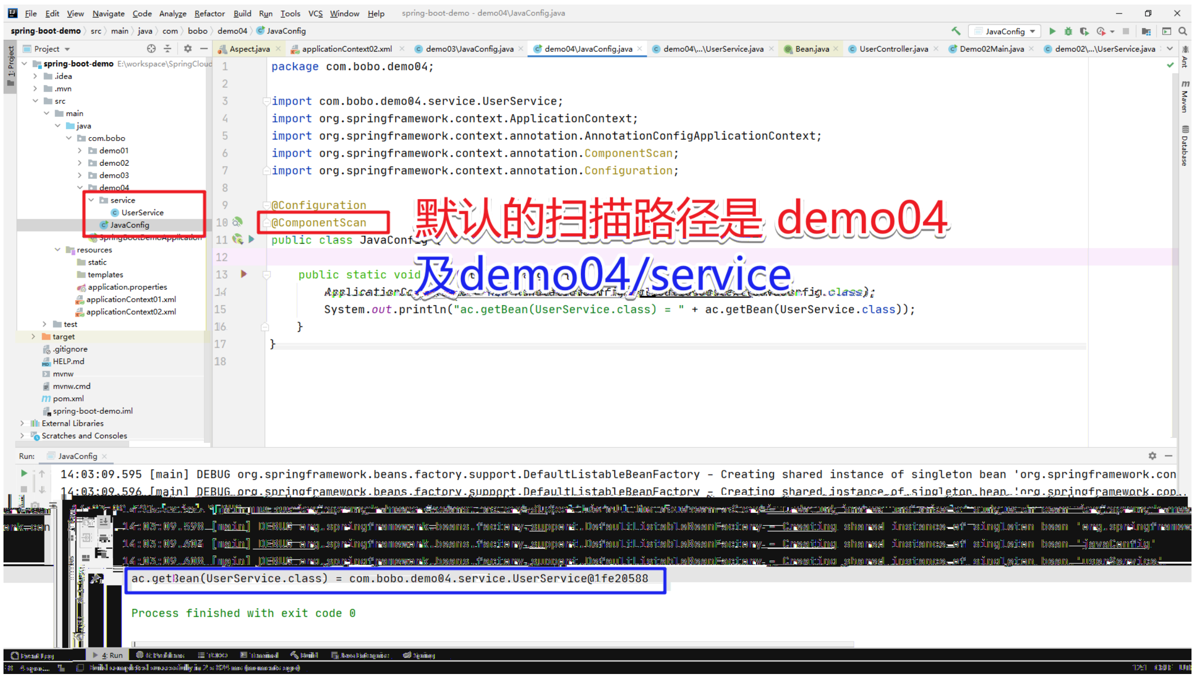
@ComponentScan XML `<component-scan>`

UserService

```
@Service
public class UserService {
}
```

Java

```
@Configuration
@ComponentScan
public class JavaConfig {

    public static void main(String[] args) {
        ApplicationContext ac = new
AnnotationConfigApplicationContext(JavaConfig.class);
        System.out.println("ac.getBean(UserService.class) = " +
ac.getBean(UserService.class));
    }
}
```



```
package com.bobo.demo04;

import com.bobo.demo04.service.UserService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan
public class JavaConfig {

    public static void main(String[] args) {
        ApplicationContext ac = new AnnotationConfigApplicationContext(JavaConfig.class);
        System.out.println("ac.getBean(UserService.class) = " + ac.getBean(UserService.class));
    }
}
```

```
14:03:09.595 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.con
14:03:09.596 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.con
14:03:09.598 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframew
14:03:09.604 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'javaConfig'
14:03:09.608 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'userService'
ac.getBean(UserService.class) = com.bobo.demo04.service.UserService@1fe20588

Process finished with exit code 0
```

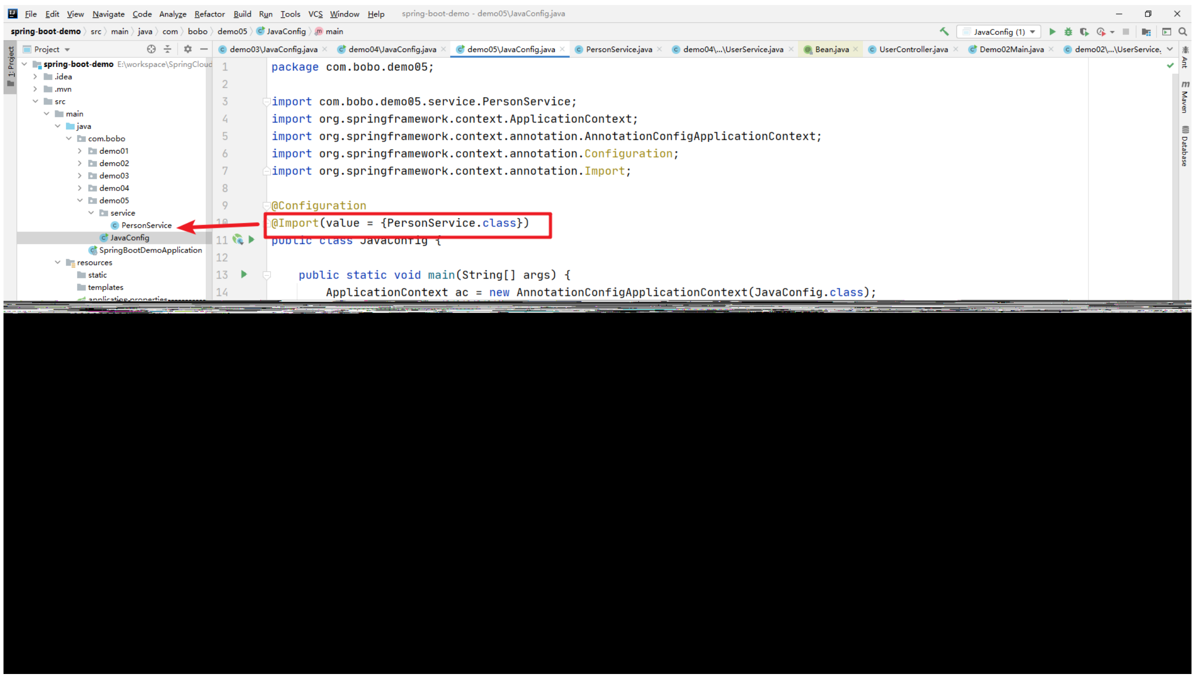
```
@Configuration
// 指定特定的扫描路径
@ComponentScan(value = {"com.bobo.demo04"})
public class JavaConfig {

    public static void main(String[] args) {
        ApplicationContext ac = new
AnnotationConfigApplicationContext(JavaConfig.class);
        System.out.println("ac.getBean(UserService.class) = " +
ac.getBean(UserService.class));
    }
}
```

@Import

@Import Spring IoC IoC
`@Bean` ,@Import

静态导入

IoC



```java
package com.bobo.demo05;

import com.bobo.demo05.service.PersonService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Import;

@Configuration
@Import(value = {PersonService.class})
public class JavaConfig {

    public static void main(String[] args) {
        ApplicationContext ac = new AnnotationConfigApplicationContext(JavaConfig.class);
```

ImportSelector

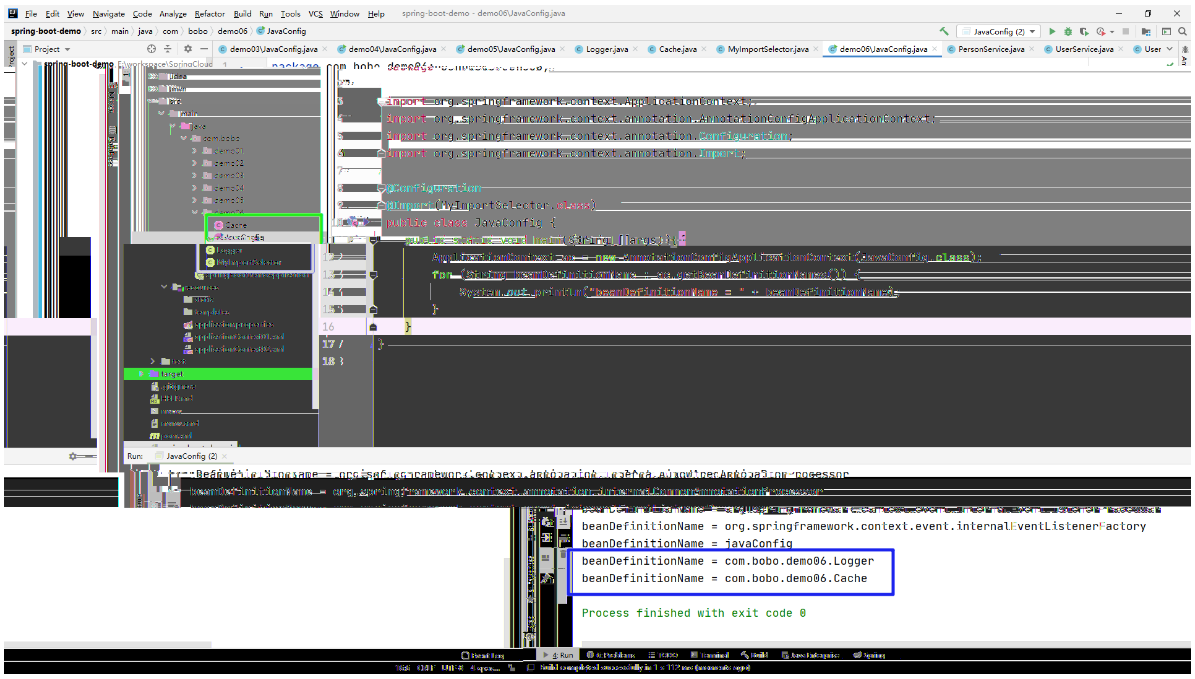
`@Import` `ImportSelector`
IoC `ImportSelector` `selectImports`

```
public class Cache {
}
public class Logger {
}
```

ImportSelector , IoC

```
public class MyImportSelector implements ImportSelector {
    @Override
    public String[] selectImports(AnnotationMetadata importingClassMetadata) {
        return new String[]{Logger.class.getName(),Cache.class.getName()};
    }
}
```

```
@Configuration
@Import(MyImportSelector.class)
public class JavaConfig {
    public static void main(String[] args) {
        ApplicationContext ac = new 
AnnotationConfigApplicationContext(JavaConfig.class);
        for (String beanDefinitionName : ac.getBeanDefinitionNames()) {
            System.out.println("beanDefinitionName = " + beanDefinitionName);
        }
    }
}
```



```
package com.bobo.demo06;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Import;

@Configuration
@Import(MyImportSelector.class)
public class JavaConfig {
    public static void main(String[] args){
        ApplicationContext ac = new AnnotationConfigApplicationContext(JavaConfig.class);
        for (String beanDefinitionName : ac.getBeanDefinitionNames()) {
            System.out.println("beanDefinitionName = " + beanDefinitionName);
        }
    }
}
```

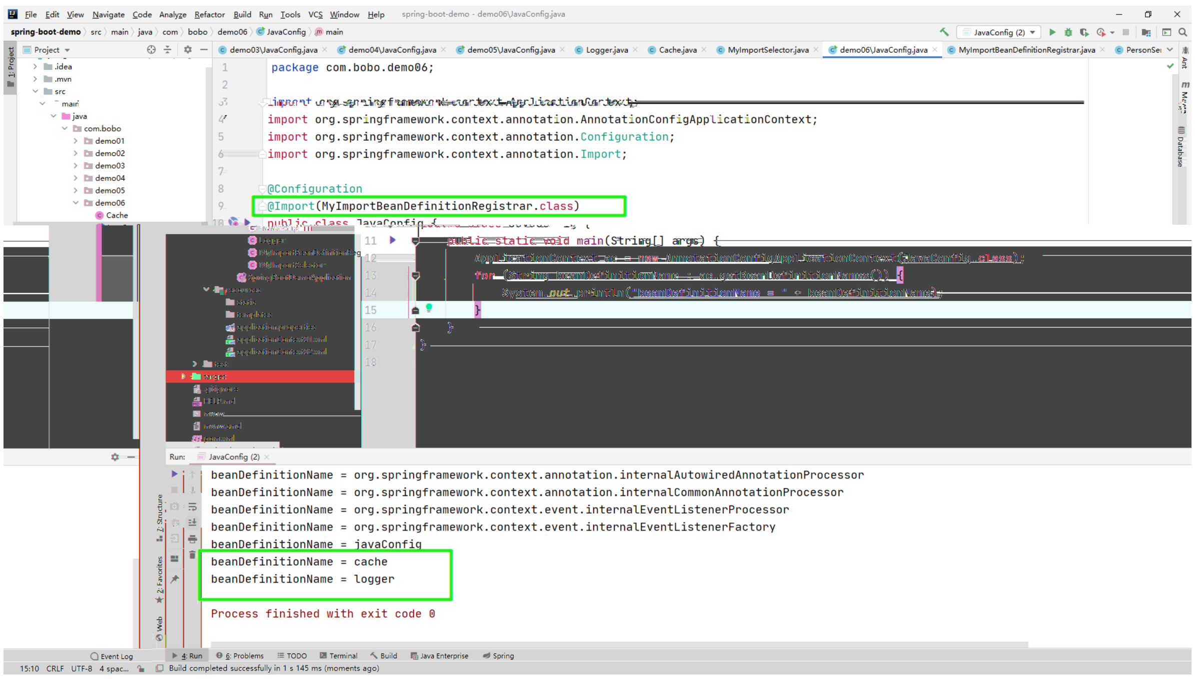
```
beanDefinitionName = org.springframework.context.annotation.internalAutowiredAnnotationProcessor
beanDefinitionName = org.springframework.context.annotation.internalCommonAnnotationProcessor
beanDefinitionName = org.springframework.context.event.internalEventListenerFactory
beanDefinitionName = javaConfig
beanDefinitionName = com.bobo.demo06.Logger
beanDefinitionName = com.bobo.demo06.Cache

Process finished with exit code 0
```

ImportBeanDefinitionRegistrar

ImportSelector `ImportBeanDefinitionRegistrar` , `ImportSelector` ,ImportBeanDefinitionRegistrar `BeanDefinitionRegistry` ,

```
public class MyImportBeanDefinitionRegistrar implements 
ImportBeanDefinitionRegistrar {
    @Override
    public void registerBeanDefinitions(AnnotationMetadata 
importingClassMetadata, BeanDefinitionRegistry registry) {
        // 将需要注册的对象封装为 RootBeanDefinition 对象
        RootBeanDefinition cache = new RootBeanDefinition(Cache.class);
        registry.registerBeanDefinition("cache",cache);

        RootBeanDefinition logger = new RootBeanDefinition(Logger.class);
        registry.registerBeanDefinition("logger",logger);
    }
}
```

```java
@Configuration
@Import(MyImportBeanDefinitionRegistrar.class)
public class JavaConfig {
    public static void main(String[] args) {
        ApplicationContext ac = new
AnnotationConfigApplicationContext(JavaConfig.class);
        for (String beanDefinitionName : ac.getBeanDefinitionNames()) {
            System.out.println("beanDefinitionName = " + beanDefinitionName);
        }
    }
}
```



```java
package com.bobo.demo06;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Import;

@Configuration
@Import(MyImportBeanDefinitionRegistrar.class)
public class JavaConfig {
    public static void main(String[] args) {
        ApplicationContext ac = new AnnotationConfigApplicationContext(JavaConfig.class);
        for (String beanDefinitionName : ac.getBeanDefinitionNames()) {
            System.out.println("beanDefinitionName = " + beanDefinitionName);
        }
    }
}
```

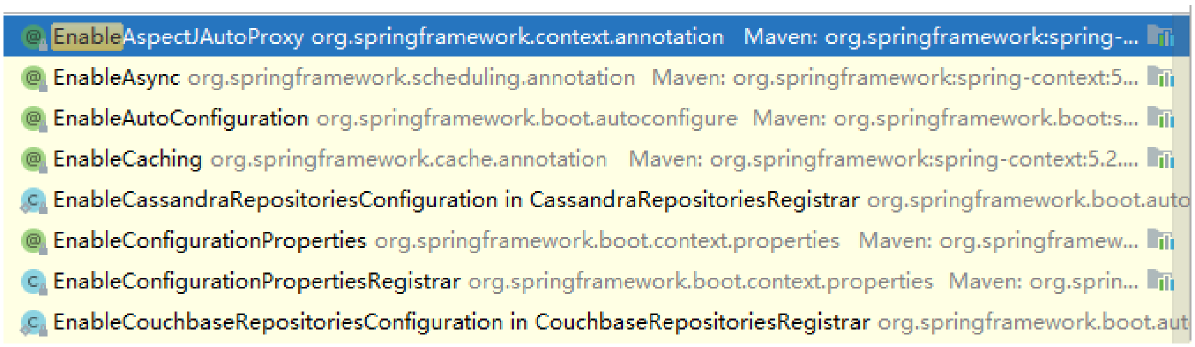
```
beanDefinitionName = org.springframework.context.annotation.internalAutowiredAnnotationProcessor
beanDefinitionName = org.springframework.context.annotation.internalCommonAnnotationProcessor
beanDefinitionName = org.springframework.context.event.internalEventListenerProcessor
beanDefinitionName = org.springframework.context.event.internalEventListenerFactory
beanDefinitionName = javaConfig
beanDefinitionName = cache
beanDefinitionName = logger

Process finished with exit code 0
```

Build completed successfully in 1 s 145 ms (moments ago)

@EnableXXX

@Enable 帮我们 Web MVC

AspectJ Caching



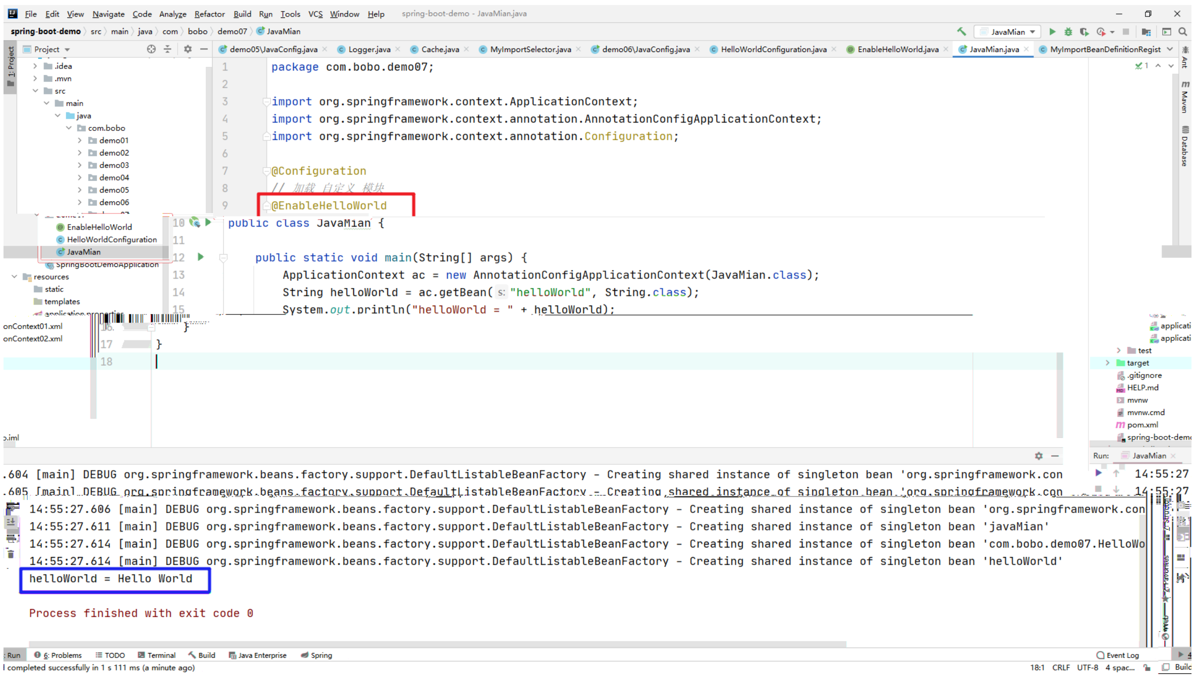
EnableAspectJAutoProxy org.springframework.context.annotation Maven: org.springframework:spring-...
EnableAsync org.springframework.scheduling.annotation Maven: org.springframework:spring-context:5...
EnableAutoConfiguration org.springframework.boot.autoconfigure Maven: org.springframework.boot:s...
EnableCaching org.springframework.cache.annotation Maven: org.springframework:spring-context:5.2....
EnableCassandraRepositoriesConfiguration in CassandraRepositoriesRegistrar org.springframework.boot.auto
EnableConfigurationProperties org.springframework.boot.context.properties Maven: org.springframew...
EnableConfigurationPropertiesRegistrar org.springframework.boot.context.properties Maven: org.sprin...
EnableCouchbaseRepositoriesConfiguration in CouchbaseRepositoriesRegistrar org.springframework.boot.aut

```
/**
 * 定义一个Java配置类
 */
@Configuration
public class HelloWorldConfiguration {

    @Bean
    public String helloWorld(){
        return "Hello World";
    }
}
```

@Enable

```
/**
 * 定义@Enable注解
 * 在该注解中通过 @Import 注解导入我们自定义的模块，使之生效。
 */
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Import(HelloWorldConfiguration.class)
public @interface EnableHelloWorld {
}
```

```
@Configuration
// 加载 自定义 模块
@EnableHelloWorld
public class JavaMian {

    public static void main(String[] args) {
        ApplicationContext ac = new
AnnotationConfigApplicationContext(JavaMian.class);
        String helloWorld = ac.getBean("helloWorld", String.class);
        System.out.println("helloWorld = " + helloWorld);
    }
}
```



```java
package com.bobo.demo07;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Configuration;

@Configuration
// 加载 自定义 模块
@EnableHelloWorld
public class JavaMian {

    public static void main(String[] args) {
        ApplicationContext ac = new AnnotationConfigApplicationContext(JavaMian.class);
        String helloWorld = ac.getBean( s: "helloWorld", String.class);
        System.out.println("helloWorld = " + helloWorld);
    }
}
```

```
.604 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.con
.605 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.con
14:55:27.606 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.con
14:55:27.611 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'javaMian'
14:55:27.614 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'com.bobo.demo07.HelloWo
14:55:27.614 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'helloWorld'
helloWorld = Hello World

Process finished with exit code 0
```

spring-boot-demo - JavaMian.java
EnableHelloWorld
HelloWorldConfiguration
JavaMian
Run: JavaMian
completed successfully in 1 s 111 ms (a minute ago)

1.4 Spring 4.x

2013 11 1 Spring 4.0 Java8.
@Conditional @Conditional
Bean

@Conditional

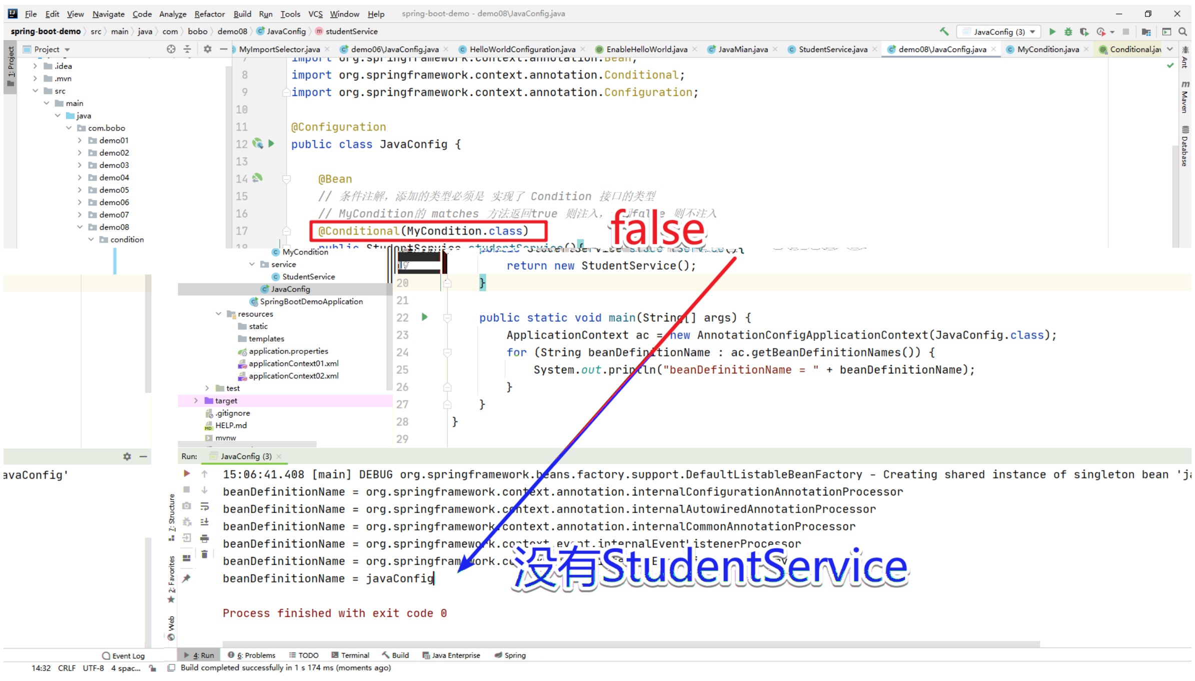
```java
// 该注解可以在 类和方法中使用
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Conditional {

    /**
     * 注解中添加的类型必须是 实现了 Condition 接口的类型
     */
    Class<? extends Condition>[] value();

}
```

Condition matches true bean false

```java
/**
 * 定义一个 Condition 接口的是实现
 */
public class MyCondition implements Condition {
    @Override
    public boolean matches(ConditionContext context, AnnotatedTypeMetadata
metadata) {
        return false; // 默认返回false
    }
}
```

Java

```java
@Configuration
public class JavaConfig {

    @Bean
    // 条件注解，添加的类型必须是 实现了 Condition 接口的类型
    // MyCondition的 matches 方法返回true 则注入，返回false 则不注入
    @Conditional(MyCondition.class)
    public StudentService studentService(){
        return new StudentService();
    }

    public static void main(String[] args) {
        ApplicationContext ac = new
AnnotationConfigApplicationContext(JavaConfig.class);
        for (String beanDefinitionName : ac.getBeanDefinitionNames()) {
            System.out.println("beanDefinitionName = " + beanDefinitionName);
        }
    }
}
```

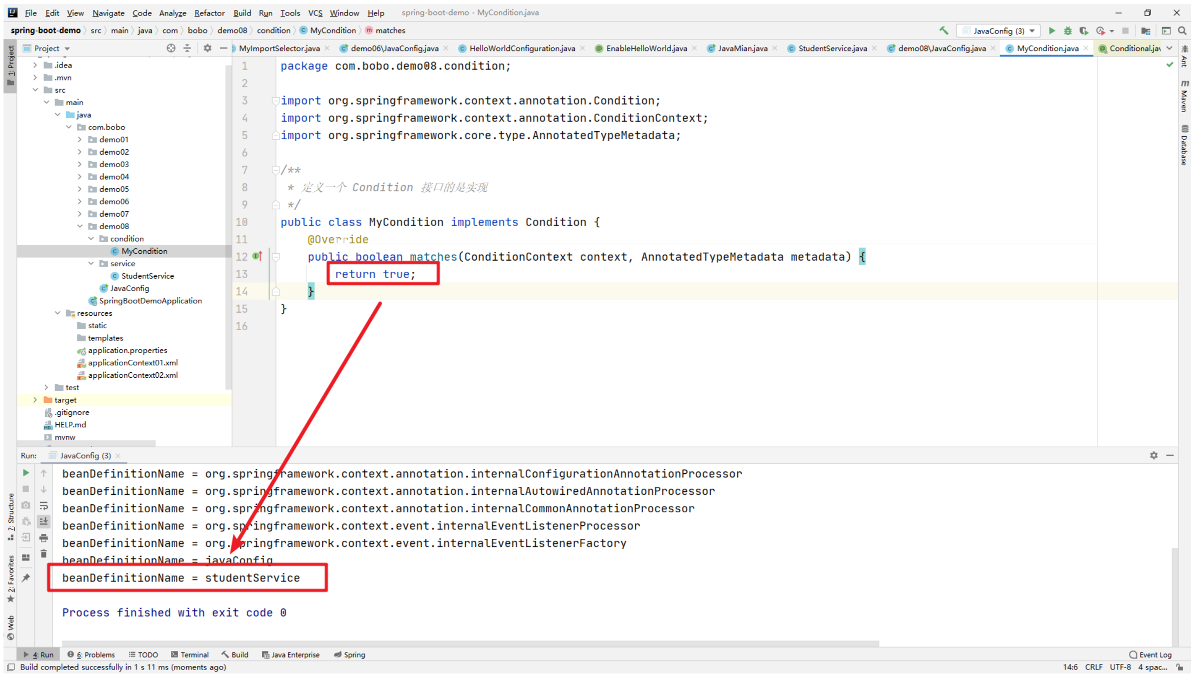


```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Conditional;
import org.springframework.context.annotation.Configuration;

@Configuration
public class JavaConfig {

    @Bean
    // 条件注解，添加的类型必须是 实现了 Condition 接口的类型
    // MyCondition的 matches 方法返回true 则注入，返回false 则不注入
    @Conditional(MyCondition.class)
    public StudentService studentService(){
        return new StudentService();
    }

    public static void main(String[] args) {
        ApplicationContext ac = new AnnotationConfigApplicationContext(JavaConfig.class);
        for (String beanDefinitionName : ac.getBeanDefinitionNames()) {
            System.out.println("beanDefinitionName = " + beanDefinitionName);
        }
    }
}
```

```
15:06:41.408 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'ja
beanDefinitionName = org.springframework.context.annotation.internalConfigurationAnnotationProcessor
beanDefinitionName = org.springframework.context.annotation.internalAutowiredAnnotationProcessor
beanDefinitionName = org.springframework.context.annotation.internalCommonAnnotationProcessor
beanDefinitionName = org.springframework.context.event.internalEventListenerProcessor
beanDefinitionName = org.springframework.context.event.internalEventListenerFactory
beanDefinitionName = javaConfig

Process finished with exit code 0
```

Build completed successfully in 1 s 174 ms (moments ago)

matchs true



```java
package com.bobo.demo08.condition;

import org.springframework.context.annotation.Condition;
import org.springframework.context.annotation.ConditionContext;
import org.springframework.core.type.AnnotatedTypeMetadata;

/**
 * 定义一个 Condition 接口的是实现
 */
public class MyCondition implements Condition {
    @Override
    public boolean matches(ConditionContext context, AnnotatedTypeMetadata metadata) {
        return true;
    }
}
```

```
beanDefinitionName = org.springframework.context.annotation.internalConfigurationAnnotationProcessor
beanDefinitionName = org.springframework.context.annotation.internalAutowiredAnnotationProcessor
beanDefinitionName = org.springframework.context.annotation.internalCommonAnnotationProcessor
beanDefinitionName = org.springframework.context.event.internalEventListenerProcessor
beanDefinitionName = org.springframework.context.event.internalEventListenerFactory
beanDefinitionName = javaConfig
beanDefinitionName = studentService

Process finished with exit code 0
```

Build completed successfully in 1 s 11 ms (moments ago)

@Conditional IoC SpringBoot 4.x `@EventListener`, ApplicationListener ,`@AliasFor` `@CrossOrigin`

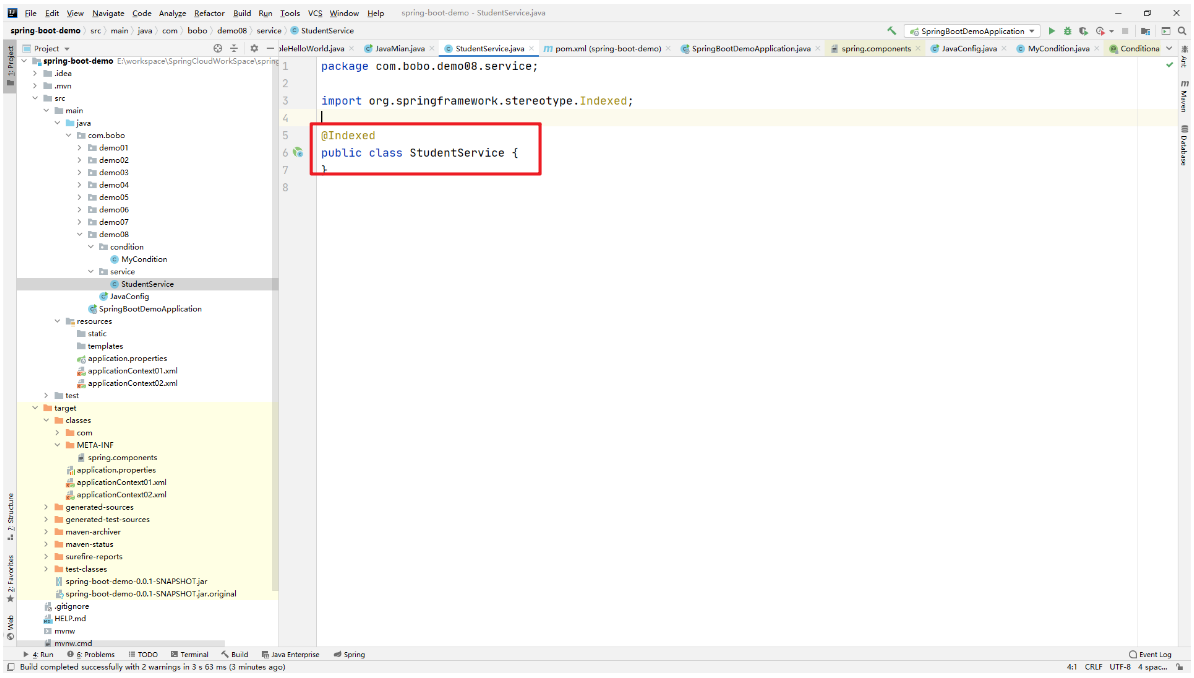
1.5 Spring 5.x

2017 9 28 Spring 5.0 5.0 SpringBoot2.0 Spring Boot @ComponentScan Spring 5.0 **@Indexed** Spring

`@Indexed` `META-INT/spring.components` Spring `ComponentScan` `META-INT/spring.components` `CandidateComponentsIndexLoader` `CandidateComponentsIndex` `@ComponentScan` package `CandidateComponentsIndex`

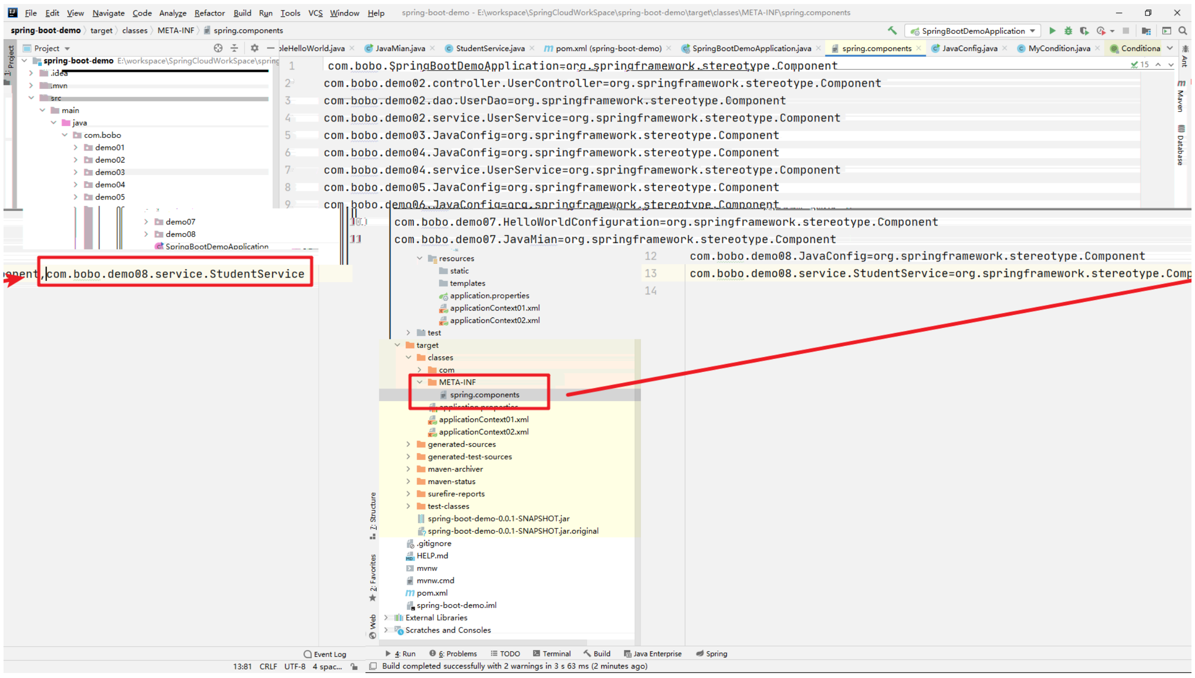
```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context-indexer</artifactId>
</dependency>
```

@Indexed



```java
package com.bobo.demo08.service;

import org.springframework.stereotype.Indexed;

@Indexed
public class StudentService {
}
```

Build completed successfully with 2 warnings in 3 s 63 ms (3 minutes ago)



spring-boot-demo - E:\workspace\SpringCloudWorkSpace\spring-boot-demo\target\classes\META-INF\spring.components

```
com.bobo.SpringBootDemoApplication=org.springframework.stereotype.Component
com.bobo.demo02.controller.UserController=org.springframework.stereotype.Component
com.bobo.demo02.dao.UserDao=org.springframework.stereotype.Component
com.bobo.demo02.service.UserService=org.springframework.stereotype.Component
com.bobo.demo03.JavaConfig=org.springframework.stereotype.Component
com.bobo.demo04.JavaConfig=org.springframework.stereotype.Component
com.bobo.demo04.service.UserService=org.springframework.stereotype.Component
com.bobo.demo05.JavaConfig=org.springframework.stereotype.Component
com.bobo.demo06.JavaConfig=org.springframework.stereotype.Component
com.bobo.demo07.HelloWorldConfiguration=org.springframework.stereotype.Component
com.bobo.demo07.JavaMian=org.springframework.stereotype.Component
com.bobo.demo08.JavaConfig=org.springframework.stereotype.Component
com.bobo.demo08.service.StudentService=org.springframework.stereotype.Comp
```

```
onent,com.bobo.demo08.service.StudentService
```

Build completed successfully with 2 warnings in 3 s 63 ms (2 minutes ago)

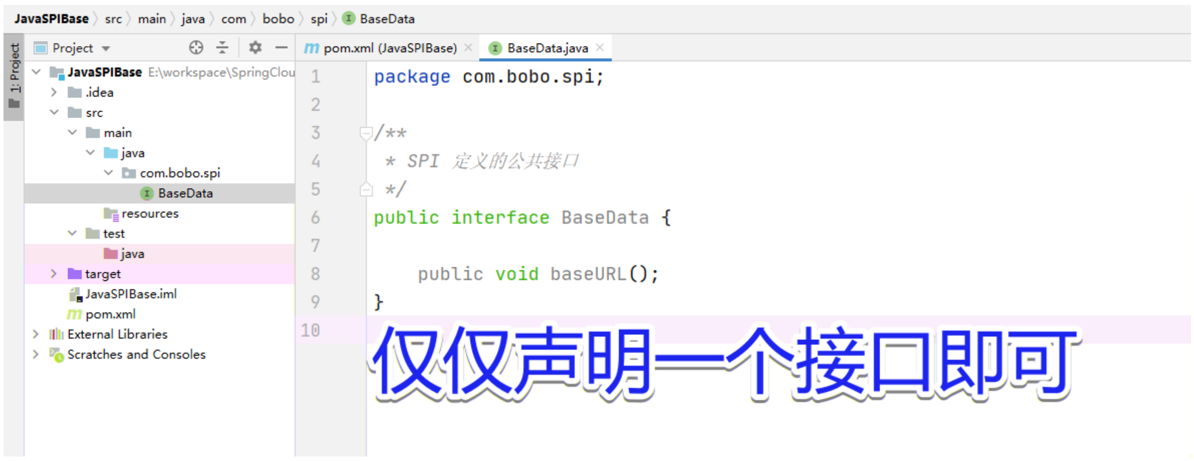
2. 什么是SPI

SPI SpringBoot SPI
SpringBoot

SPI Service Provider Interface ClassPath
META-INF/services

Dubbo JDBC SPI

案例介绍

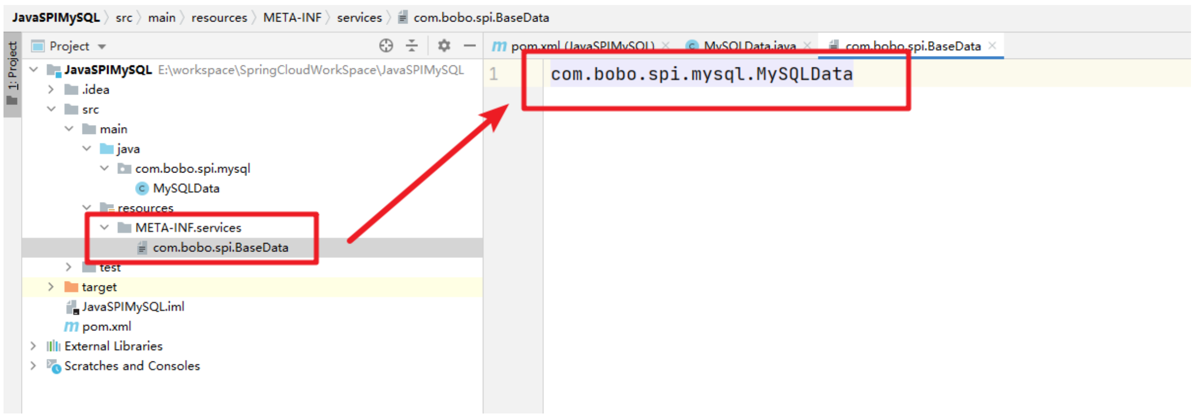


```
package com.bobo.spi;

/**
 * SPI 定义的公共接口
 */
public interface BaseData {

    public void baseURL();
}
```

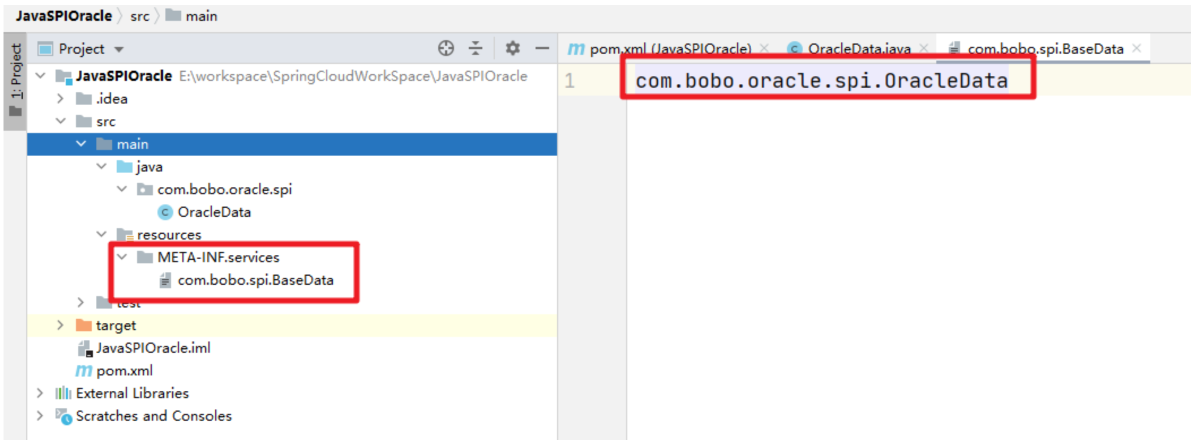
```
    <dependencies>
        <dependency>
            <groupId>com.bobo</groupId>
            <artifactId>JavaSPIBase</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

```
/**
 * SPI：MySQL对于 baseURL 的一种实现
 */
public class MySQLData implements BaseData {
    @Override
    public void baseURL() {
        System.out.println("mysql 的扩展实现....");
    }
}
```

resources META-INF/services

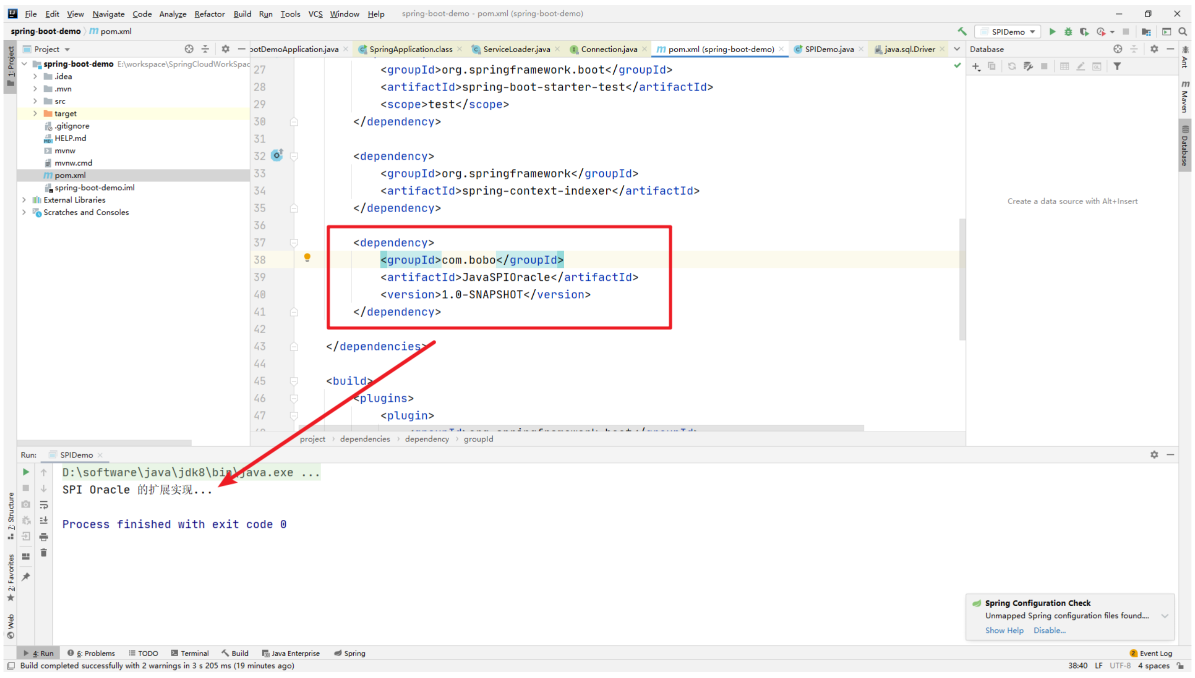


```
com.bobo.spi.mysql.MySQLData
```



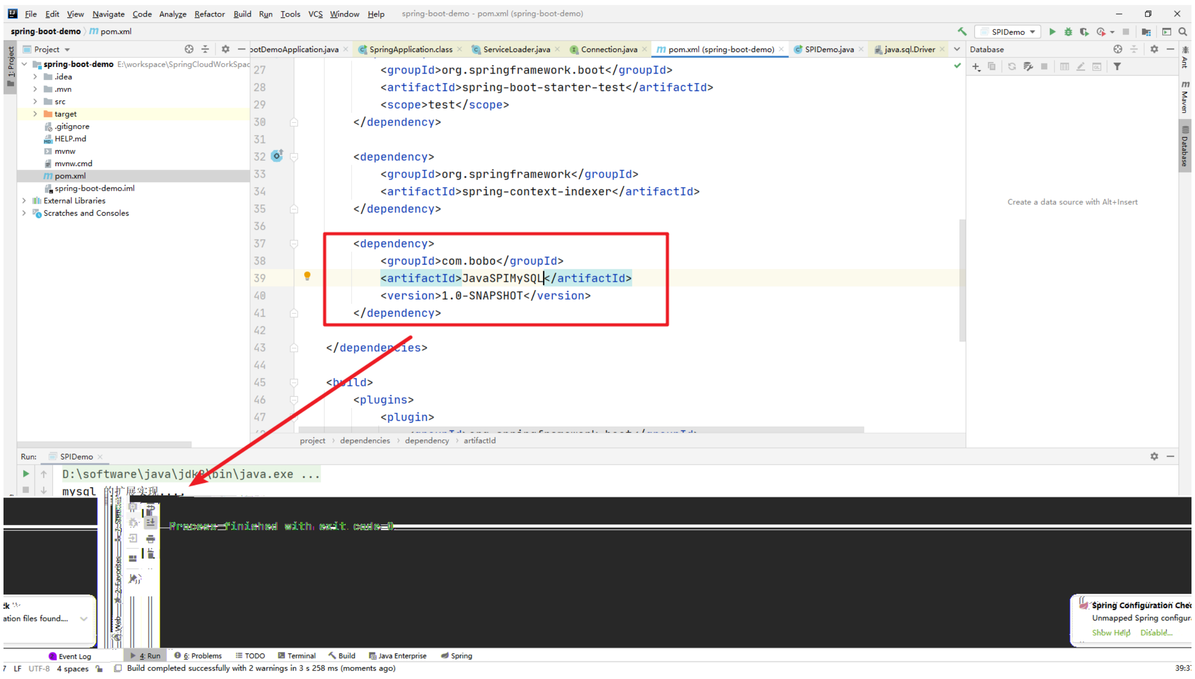
```
com.bobo.oracle.spi.OracleData
```

```
public static void main(String[] args) {
    ServiceLoader<BaseData> providers = ServiceLoader.load(BaseData.class);
    Iterator<BaseData> iterator = providers.iterator();
    while(iterator.hasNext()){
        BaseData next = iterator.next();
        next.baseURL();
    }
}
```



```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-indexer</artifactId>
        </dependency>

        <dependency>
            <groupId>com.bobo</groupId>
            <artifactId>JavaSPIOracle</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
```

```
D:\software\java\jdk8\bin\java.exe ...
SPI Oracle 的扩展实现...

Process finished with exit code 0
```



```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-indexer</artifactId>
    </dependency>

    <dependency>
        <groupId>com.bobo</groupId>
        <artifactId>JavaSPIMySQL</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
```

```
D:\software\java\jdk8\bin\java.exe ...
mysql 的扩展实现....

Process finished with exit code 0
```

```
Build completed successfully with 2 warnings in 3 s 258 ms (moments ago)
```

源码查看

ServiceLoader

ServiceLoader

```
    // 配置文件的路径
    private static final String PREFIX = "META-INF/services/";

    // 加载的服务  类或者接口
    private final Class<S> service;

    // 类加载器
    private final ClassLoader loader;

    // 访问权限的上下文对象
    private final AccessControlContext acc;

    // 保存已经加载的服务类
    private LinkedHashMap<String,S> providers = new LinkedHashMap<>();

    // 内部类，真正加载服务类
    private LazyIterator lookupIterator;
```

load

load LazyIterator

```
public final class ServiceLoader<S> implements Iterable<S>
    private ServiceLoader(Class<S> svc, ClassLoader cl) {
        //要加载的接口
        service = Objects.requireNonNull(svc, "Service interface cannot be
null");
        //类加载器
        loader = (cl == null) ? ClassLoader.getSystemClassLoader() : cl;
        //访问控制器
        acc = (System.getSecurityManager() != null) ?
AccessController.getContext() : null;
```

```
         reload();

    }
    public void reload() {
        //先清空
        providers.clear();
        //实例化内部类
        LazyIterator lookupIterator = new LazyIterator(service, loader);
    }
}
```

LazyIterator iterator.hasNext iterator.next LazyIterator

```
private class LazyIterator implements Iterator<S>{
    Class<S> service;
    ClassLoader loader;
    Enumeration<URL> configs = null;
    Iterator<String> pending = null;
    String nextName = null;
    [illegible] I,.Nas mee();
        //第二次调用的时候，已经解析完成了，直接返回
        [illegible]) {
      =(next[illegible];
        }/
        [illegible]{Naf <ceel,vice, loaddll{
         /  //META-INF/services/ 加上接口的全限定类名，就是文件服务类的文件
       =    //META-INF/services/com.viewscenes.netsupervisor.spi.SPIService
         /  String fullName = PREFIX + service.getName();
            //将文件路径转成URL对象
            configs = loader4Ric It+ nmeI.,etNae(){
```

加载文件资源

获取文件里面配置的类全限定名

next lookupIterator.nextService

根据类全限定名实例化对象

```
        providers.put(cn, p);
        return p;
    }
}
```