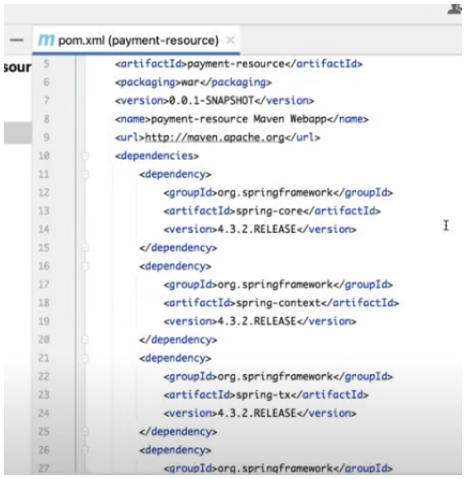
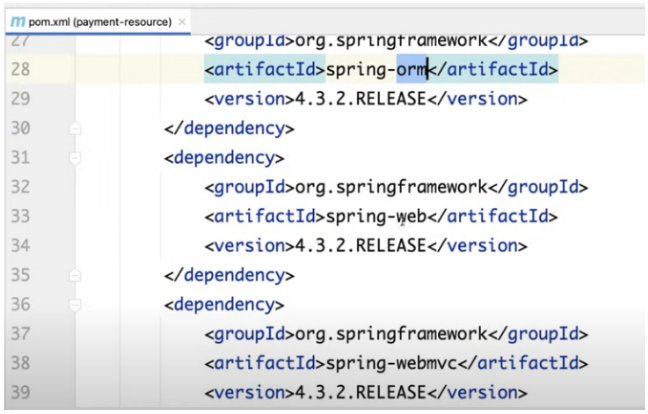
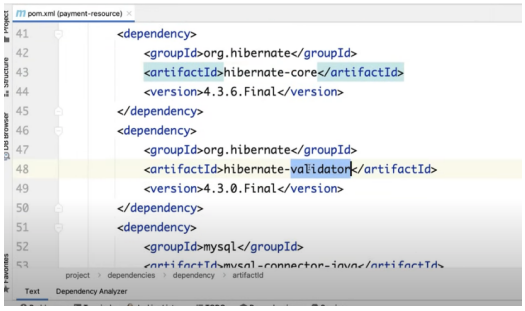
1. Why will you choose Spring Boot over Spring Framework ?

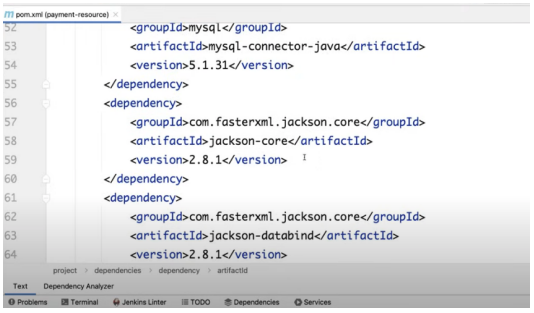
Dependency Resolution / Avoid Version conflict

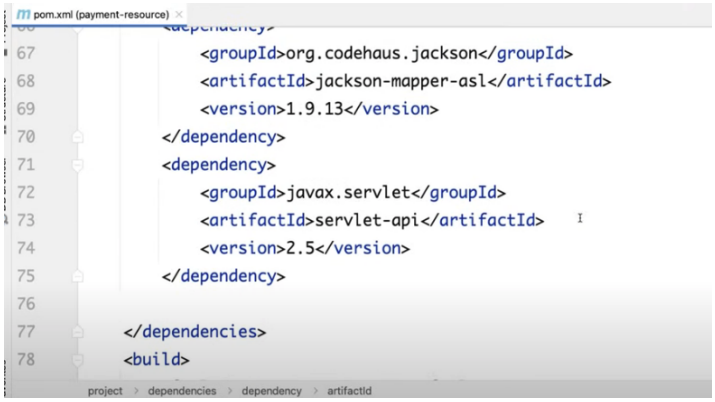
In the case of spring, we need to add a lot of dependencies manually. And also there is a chance for version conflicts because of this.



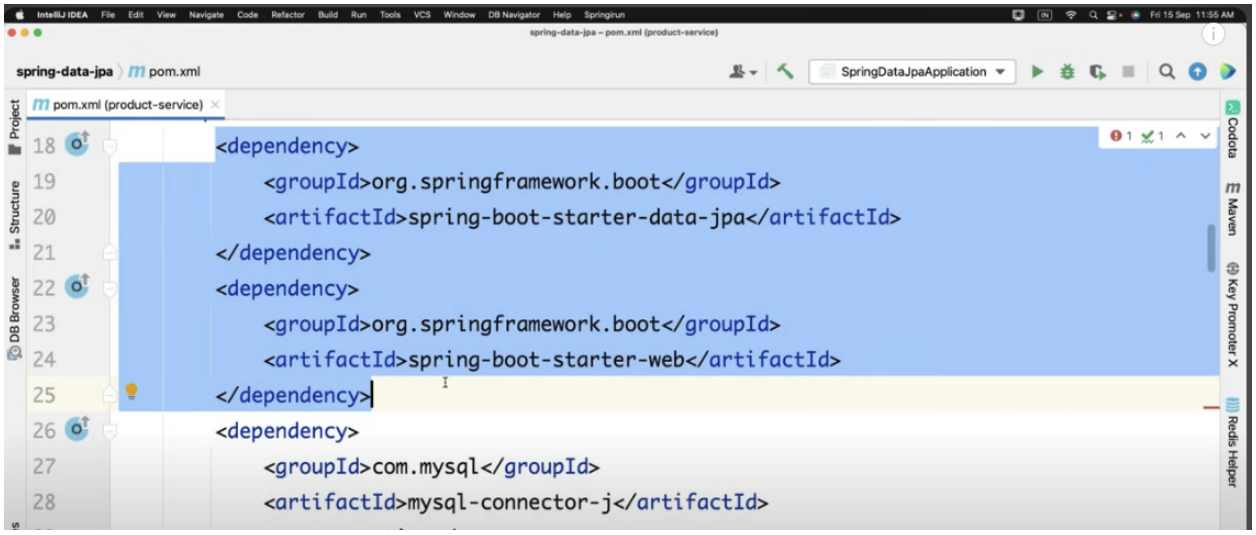






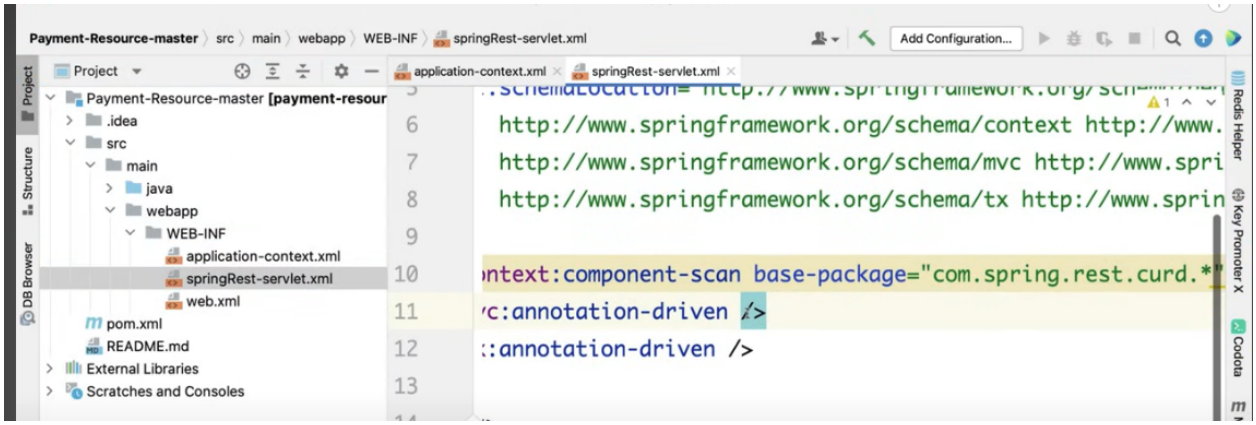


Incase of spring boot we need only below 2 Dependency.

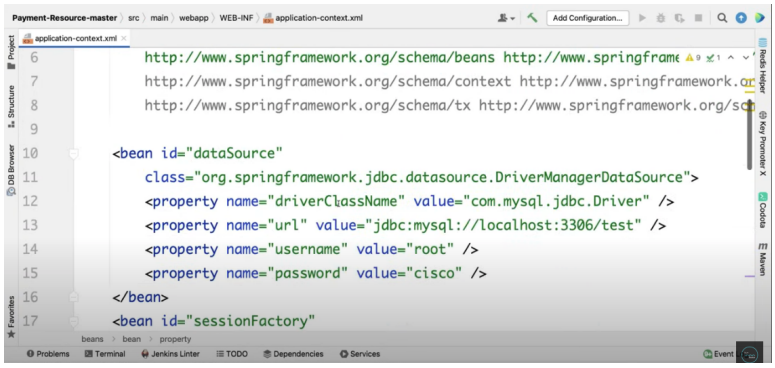


Avoid additional configuration

The below are the 3 configurations we need to perform in case of spring.



1st configuration application-context.xml. Create all the configuration related beans here



```
application-context.xml
        <property name="password" value="cisco" />
    </bean>
    <bean id="sessionFactory"
        class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="annotatedClasses">
            <list>
                <value>com.spring.rest.curd.model.Payment</value>
            </list>
        </property>
        <property name="hibernateProperties">
```

```
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</p
                <prop key="hibernet.show_sql">true</prop>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
    </bean>
```

Tag name: **hibernate.hbm2ddl.auto**
Description : The key of the property entry.
Maven: org.springframework:spring-beans:4.3.2.RELEASE
`xsd:attribute` on www.w3.org

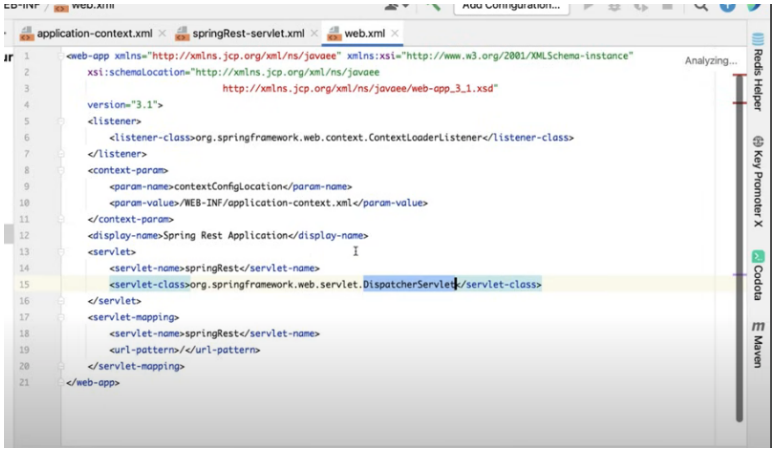
```
    <bean id="transactionManager"
        class="org.springframework.orm.hibernate4.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
</beans>
```

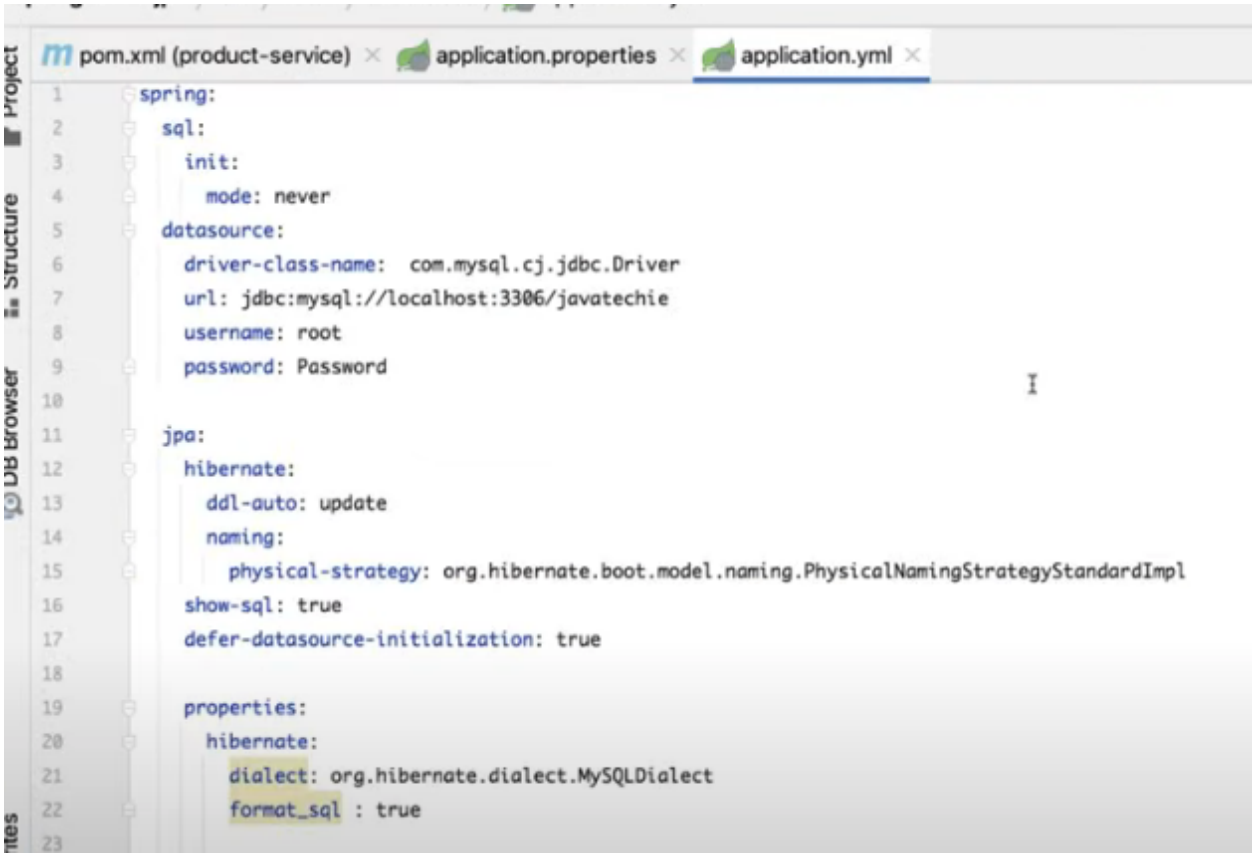
2nd configuration springRest-servlet.xml. What are all the packages it has to scan

application-context.xml | springRest-servlet.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"
    xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.x
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
        http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

    <context:component-scan base-package="com.spring.rest.curd.*"></context:component-scan>
    <mvc:annotation-driven />
    <tx:annotation-driven />

</beans>
```

In web.xml - we have to tell where is our application-context.xml dispatcher servlet that need to be used.





Embed Tomcat, Jetty (no need to deploy WAR files)

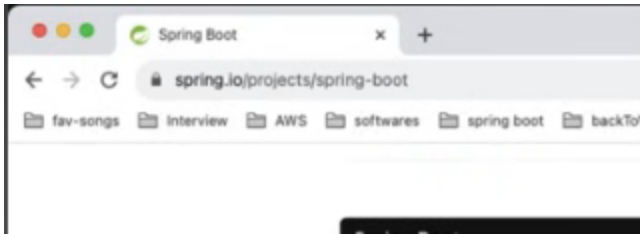
In the case of spring we have to create our jar or war and deploy into the tomcat server to start the application.
In case of spring-boot app embedded tomcat server loads target/classes folder and start the application. In other words it takes compiled class and start the application.

Provide production-ready features such as metrics, health checks

In case of spring-boot, Actuator helps with additional endpoints to monitor the health of our application. In spring it is not that straight forward.

2. What all spring boot starter you have used or what all module you have worked on ?

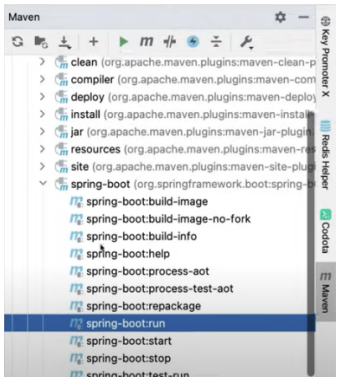
- spring boot starter web
- spring boot starter data JPA
- spring boot starter AOP
- spring boot starter web services
- spring boot starter security
- Spring Boot Starter for Apache Kafka
- Spring Boot Starter for Spring Cloud
- spring boot starter thymeleaf



This page helps how to work with spring boot starters

3. How will you run your Spring Boot application ?

```
16    public static void main(String[] args) {
17        SpringApplication.run(InterviewQaApplication.class, args);
18    }
```



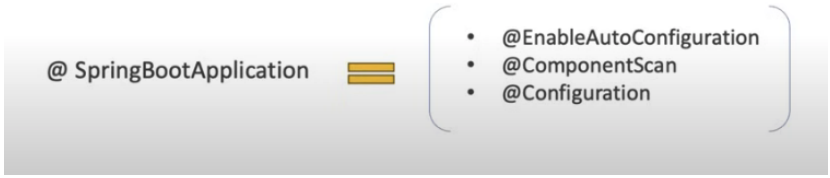
If we create jar file and extract it we find 2 folders

```
javatechie@Basantas-iMac Desktop %
javatechie@Basantas-iMac Desktop %
javatechie@Basantas-iMac Desktop % jar xf interview-qa-0.0.1-SNAPSHOT.jar
javatechie@Basantas-iMac Desktop %
```

In BOOT-INF it contains classes and lib
In META-INF contains application info in manifest.mf file.

```
Manifest-Version: 1.0
Created-By: Maven JAR Plugin 3.3.0
Build-Jdk-Spec: 17
Implementation-Title: interview-qa
Implementation-Version: 0.0.1-SNAPSHOT
Main-Class: org.springframework.boot.loader.JarLauncher
Start-Class: com.javatechie.InterviewQaApplication
Spring-Boot-Version: 3.1.2
Spring-Boot-Classes: BOOT-INF/classes/
Spring-Boot-Lib: BOOT-INF/lib/
Spring-Boot-Classpath-Index: BOOT-INF/classpath.idx
Spring-Boot-Layers-Index: BOOT-INF/layers.idx
```

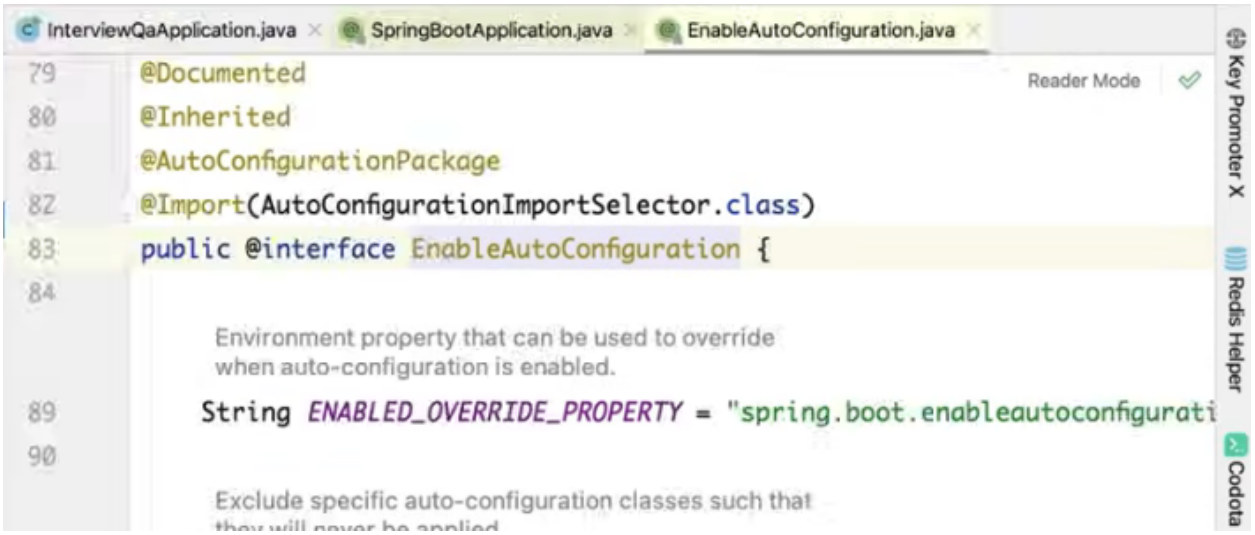
4. What is the purpose of the @SpringBootApplication annotation in a Spring Boot application ?



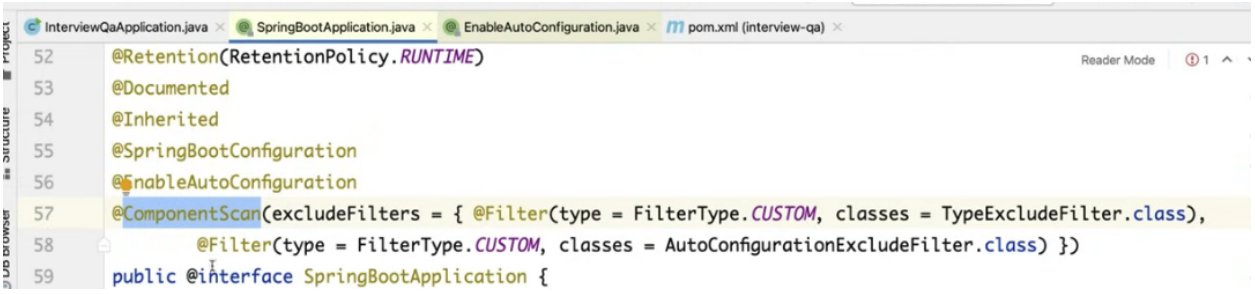
By seeing jar files and properties “EnableAutoConfiguration” enables auto configuration. Example by seeing data-jpa jar and datasource properties it enables database configuration and so on.

ComponentScan searches for all the components and beans and loads them into spring container/context.

@Configuration helps to load extra bean into context or import additional custom configuration



By seeing jar files and properties “EnableAutoConfiguration” enables configuration. Example by seeing data-jpa jar and datasource properties it enables database configuration and so on.



ComponentScan searches for all the components and beans and loads them into spring container/context. By default it will scan only in its root folder package. External to its root folder will be ignored. If we want we need explicitly specify other packages as shown below

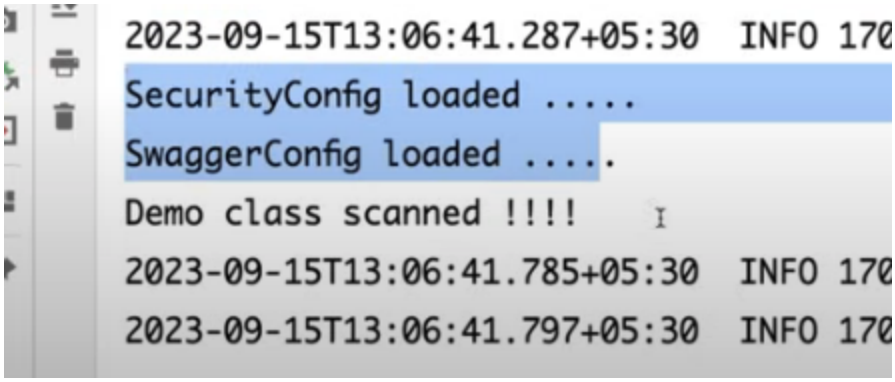
```
@SpringBootApplication(scanBasePackages = {"com.business.common","com.javatechie.*"})
public class InterviewQaApplication  {
```

@Configuration helps to load extra bean into context or import additional custom configuration

```
@Configuration
public class SwaggerConfig {

    public SwaggerConfig() {
        System.out.println("SwaggerConfig loaded .....");
    }
}
```

5. can I directly use above 3 annotation in my main class , instead of using @SpringBootApplication annotation , if yes will my application work as expected

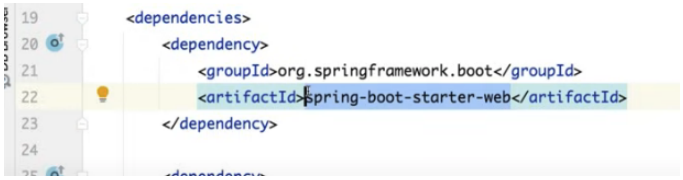
```
15
16  @EnableAutoConfiguration
17  @ComponentScan(basePackages = {"com.javatechie.*","com.business"})
18  @Import({SecurityConfig.class, SwaggerConfig.class})
19  public class InterviewQaApplication  {
20
```

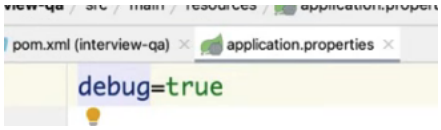


It works

6. What is Auto configuration in spring boot ?

I have only spring-boot-starter-web and add debug=true in the application.properties





We will see while running the application we see positive and negative matches

```
JacksonAutoConfiguration matched:
   - @ConditionalOnClass found required class 'com.fasterxml.jackson.databind.ObjectMapper' (OnClassCondition)

JacksonAutoConfiguration.Jackson2ObjectMapperBuilderCustomizerConfiguration matched:
   - @ConditionalOnClass found required class 'org.springframework.http.converter.json.Jackson2ObjectMapperBuilder'

JacksonAutoConfiguration.JacksonObjectMapperBuilderConfiguration matched:
```

JacksonAutoConfiguration comes under positive match because of ObjectMapper.class file exist then enable JacksonAutoConfiguraiton

```
Eddú Meléndez, Ralf Ueberfuhr
@AutoConfiguration
@ConditionalOnClass(ObjectMapper.class)
public class JacksonAutoConfiguration {

    private static final Map<?, Boolean> FEATURE_DEFAULTS;
```

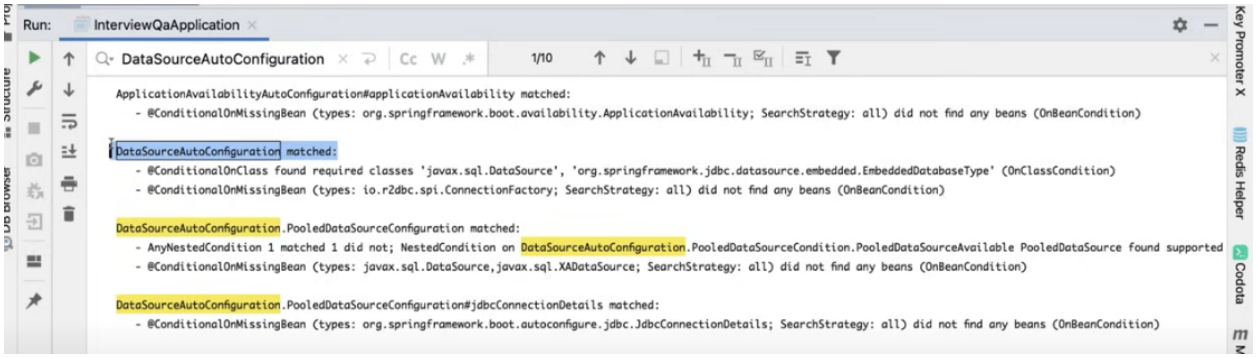
DataSourceAutoConfiguration comes under negative match because some classes are missing.

```
DataSourceAutoConfiguration:
   Did not match:
      - @ConditionalOnClass did not find required class 'org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseType

DataSourceInitializationConfiguration:
   Did not match:
      - @ConditionalOnClass did not find required class 'org.springframework.jdbc.datasource.init.DatabasePopulator' (OnCl

DataSourceTransactionManagerAutoConfiguration:
   Did not match:
      - @ConditionalOnClass did not find required class 'org.springframework.jdbc.core.JdbcTemplate' (OnClassCondition)
```

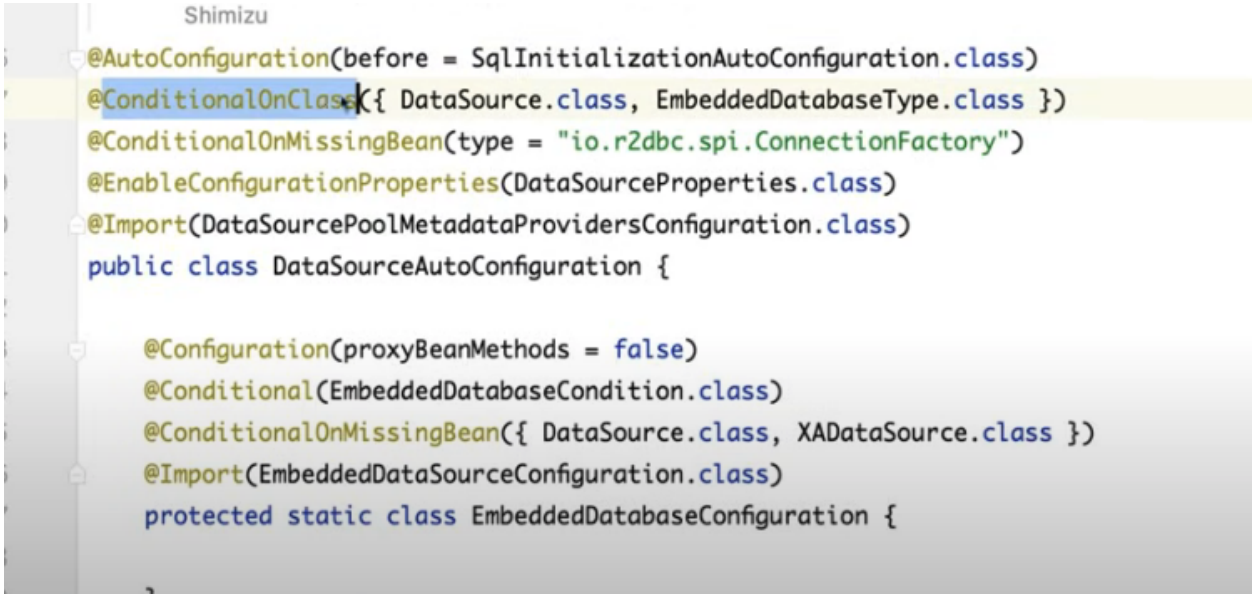
Add this dependency and try

```
34          </dependency>
35  <!--        <dependency>-->
36  <!--            <groupId>org.springframework.boot</groupId>-->
37  <!--            <artifactId>spring-boot-starter-data-jpa</artifactId>-->
38  <!--        </dependency>-->
39  <!--        <dependency>-->
40  <!--            <groupId>com.mysql</groupId>-->
41  <!--            <artifactId>mysql-connector-j</artifactId>-->
42  <!--            <scope>runtime</scope>-->
        project > dependencies
```

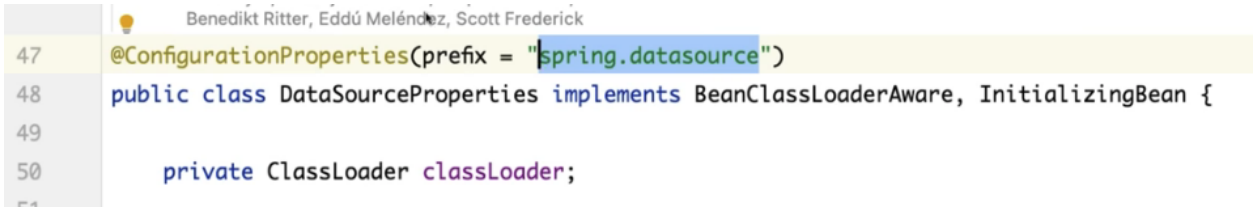
you will see DataSourceAutoConfiguration in the positive matches

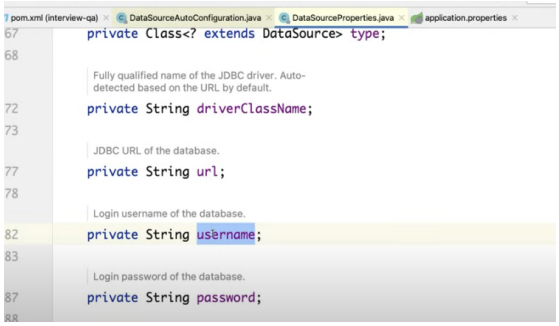


We can see what are all classes and what classes should not be in

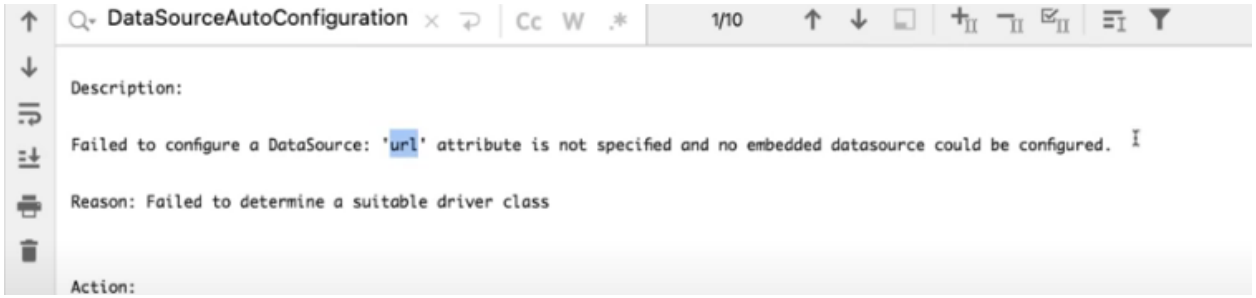


Click on DataSouceProperties.class to see what are all properties it is expecting.





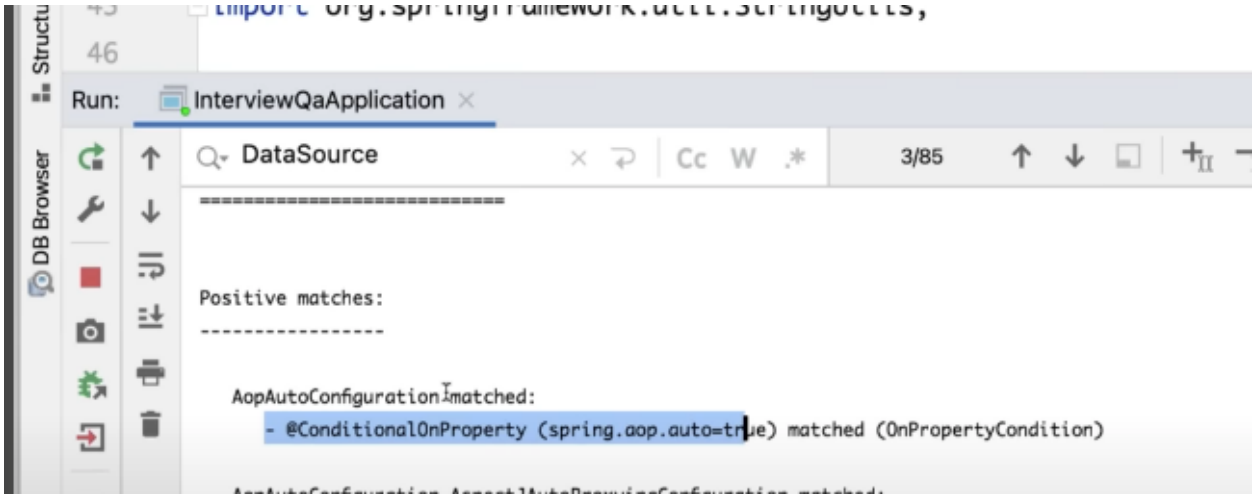
Because we have not specified any properties we get below error.



Now add these properties and try

Any single match - then it will come to a positive match. It does not mean you are done with configuration. To have complete configuration all the conditional annotations should satisfy.

Let us see another example

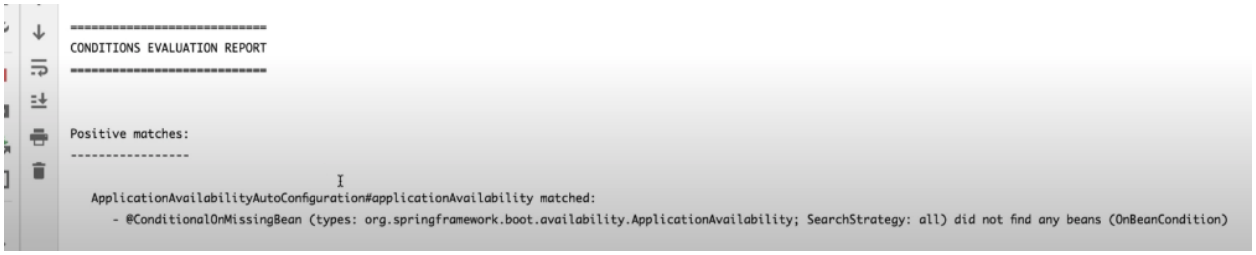


We have a positive match for AOP because of some property even though we have not that property in the application.properties file. That is because of line number 47.

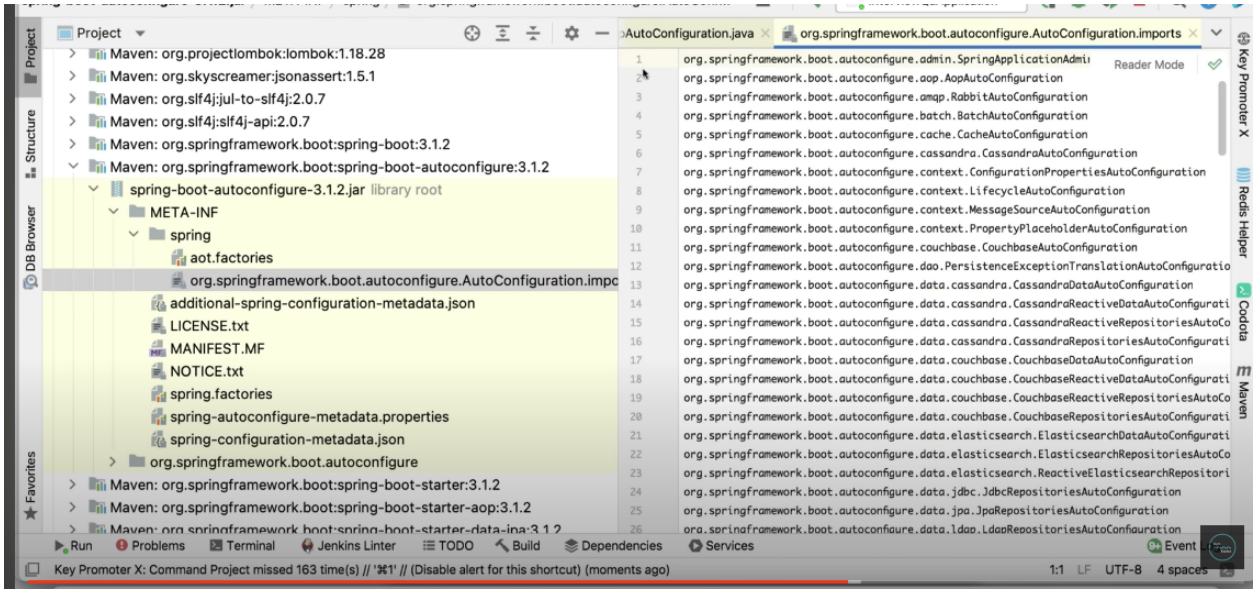
```
Author:    Dave Syer, Josh Long
@AutoConfiguration
@ConditionalOnProperty(prefix = "spring.aop", name = "auto", havingValue = "true", matchIfMissing = true)
public class AopAutoConfiguration {

    @Configuration(proxyBeanMethods = false)
    @ConditionalOnClass(Advice.class)
```

Now override that property

And start the application. Now we wont see that configuration.



In this place also we can fine all positive and negative matches



7. How can you disable a specific auto-configuration class in Spring Boot?

```
@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class,AopAutoConf
public class InterviewQaApplication  {
```

Another way to to exclude using spring.autoconfigure.exclude in the properties file.

```
11
12  spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.aop.AopAuto
```

If you have more classes to exclude use comma separated

```
11
12  e.exclude=org.springframework.boot.autoconfigure.aop.AopAutoConfiguration,
```

8. How can you customize the default configuration in Spring Boot

By overidding as shown below

InterviewQaApplication.java

```
1  server:
2    port: 8181
```

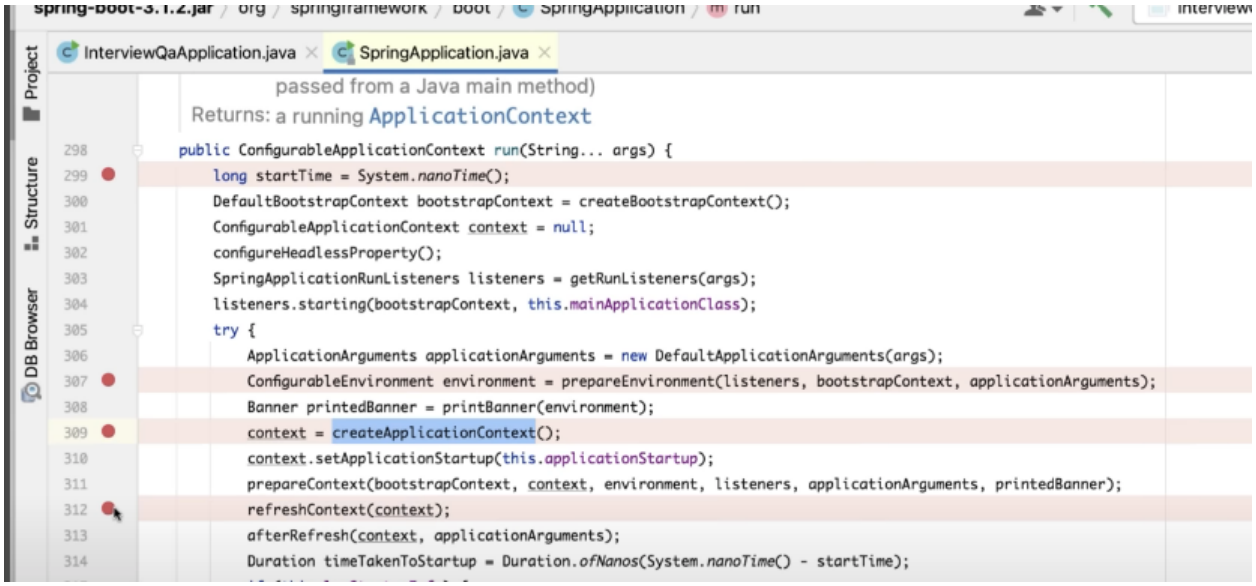
```
2
3  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
4  spring.datasource.url = jdbc:mysql://localhost:3306/javatechie
5  spring.datasource.username = root
6  spring.datasource.password = Password
7  spring.jpa.hibernate.ddl-auto = update
8  spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL
```

9. How Spring boot run() method works internally ?

➔ Create ApplicationContext
➔ Register bean into context
➔ Kicked up embedded tomcat container to run your jar/war

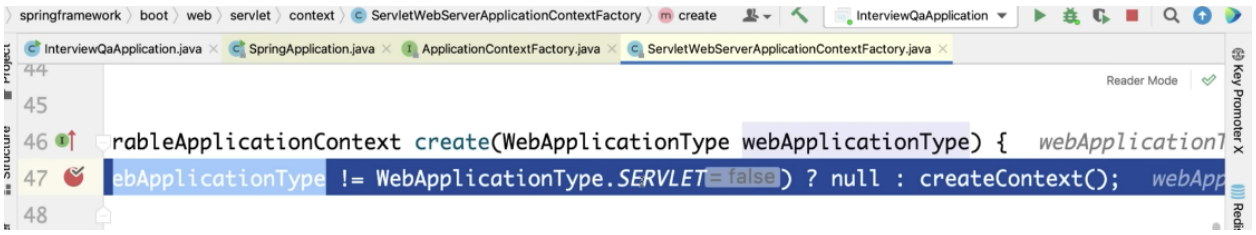
Run - it will check servlet application or reactive application or standalone application then load environment property and load the context.

Use debug to find what is happening



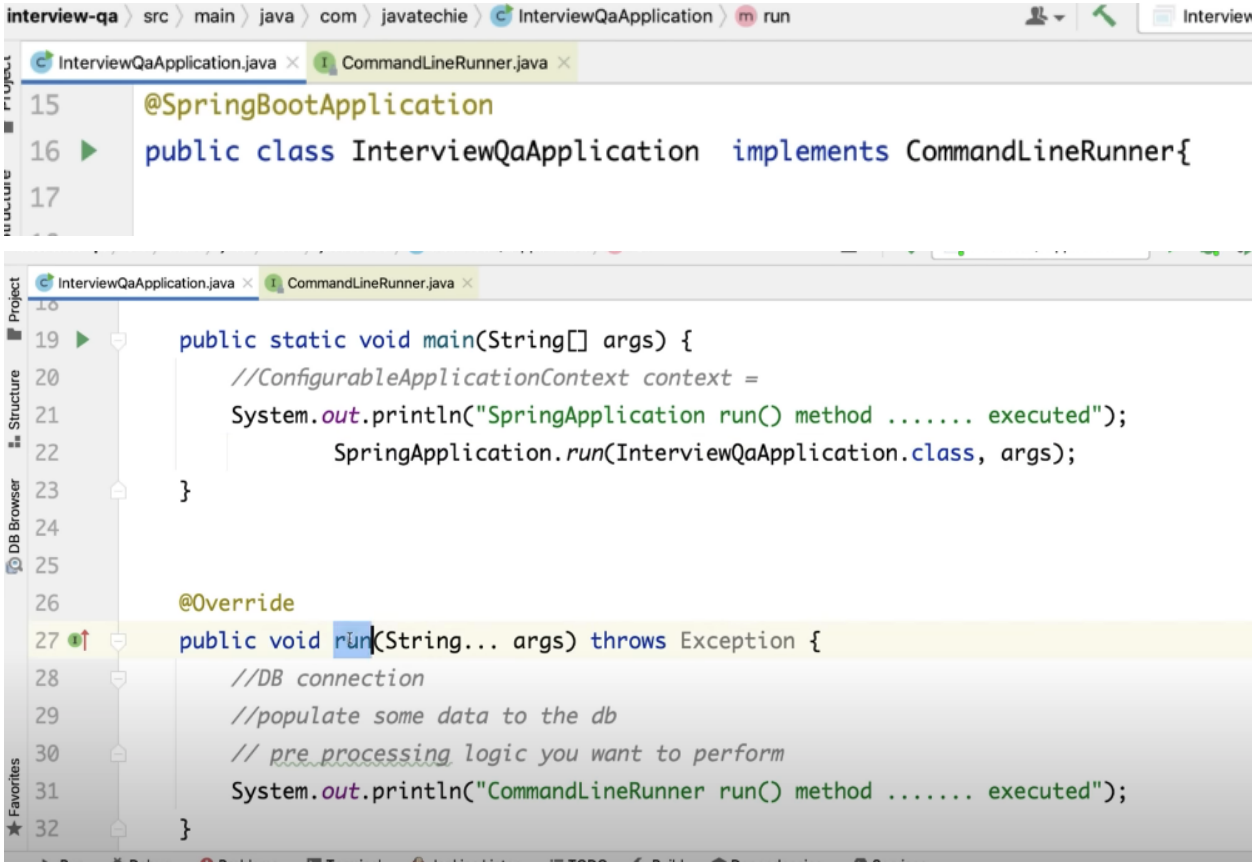
Assign run method to a variable to know what it is returning

```
8
9   public static void main(String[] args) {
0       ConfigurableApplicationContext context = SpringApplication
1               .run(InterviewQaApplication.class, args);
2   }
3
4
```



10. What is Command line runner in spring boot ?

This will help you to execute preprocessor logic during application startup.



Now we have two run methods. This is a functional interface with an abstract run method.

It executes the spring application run method first then Command line runner run method.