

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.

```
<artifactId>payment-resource</artifactId>
<packaging>war</packaging>
<version>0.0.1-SNAPSHOT</version>
<name>payment-resource Maven Webapp</name>
<url>http://maven.apache.org</url>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>4.3.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.3.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>4.3.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
    </dependency>
</dependencies>
```

```
<groupId>org.springframework</groupId>
<artifactId>spring-orm</artifactId>
<version>4.3.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>4.3.2.RELEASE</version>
</dependency>
```

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>4.3.6.Final</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>4.3.0.Final</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.31</version>
</dependency>
```

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.8.1</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.8.1</version>
</dependency>
```

```
<dependency>
    <groupId>org.codehaus.jackson</groupId>
    <artifactId>jackson-mapper-asl</artifactId>
    <version>1.9.13</version>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.5</version>
</dependency>
```

```
</dependencies>
<build>
```

Incase of spring boot we need only below 2 Dependency.

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>

```

Avoid additional configuration

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

```

<!-- SCHEMA LOCATION = http://www.springframework.org/schema/context http://www.
http://www.springframework.org/schema/mvc http://www.spri
http://www.springframework.org/schema/tx http://www.spring

<intext:component-scan base-package="com.spring.rest.cur.*"
<c:annotation-driven />
::annotation-driven />

```

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

```

<bean id="dataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/test" />
    <property name="username" value="root" />
    <property name="password" value="cisco" />
</bean>
<bean id="sessionFactory"
      beans > bean > property

```

```

<property name="username" value="root" />
<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</prop>
  <prop key="hibernate.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

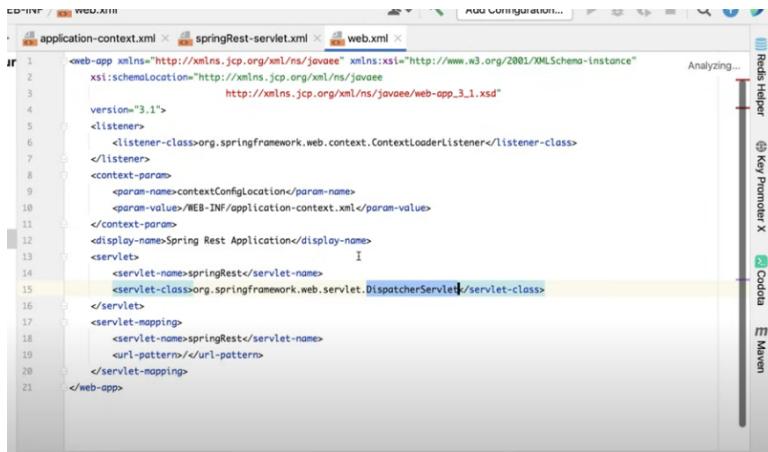
```

<?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.xsd
                           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">

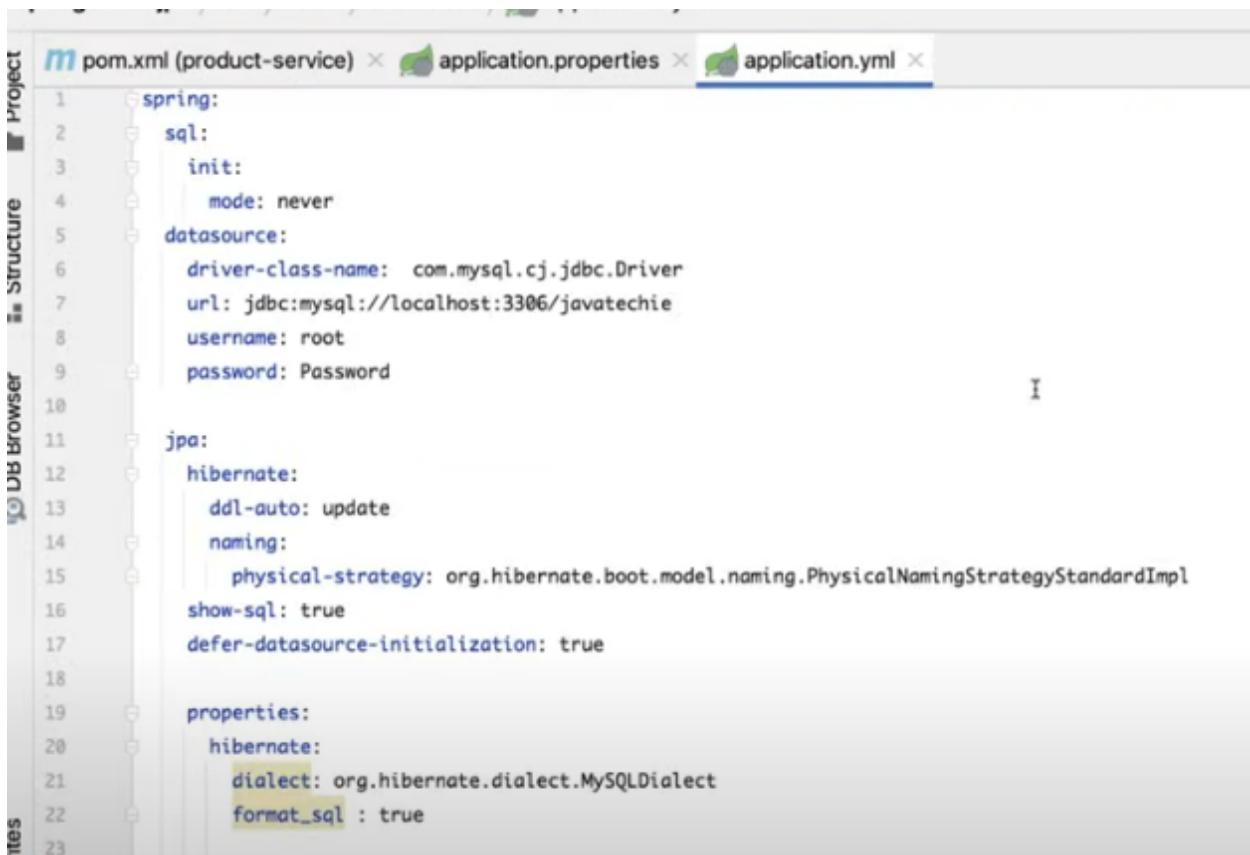
  <context:component-scan base-package="com.spring.rest.curd.*"></context:component-scan>
  <mvc:annotation-driven />
  <tx:annotation-driven />
</beans>

```

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



```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/application-context.xml</param-value>
    </context-param>
    <display-name>Spring Rest Application</display-name>
    <servlet>
        <servlet-name>springRest</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>springRest</servlet-name>
        <url-pattern>/<url-pattern>
    </servlet-mapping>
</web-app>
```



```
spring:
  sql:
    init:
      mode: never
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/javatechie
    username: root
    password: Password

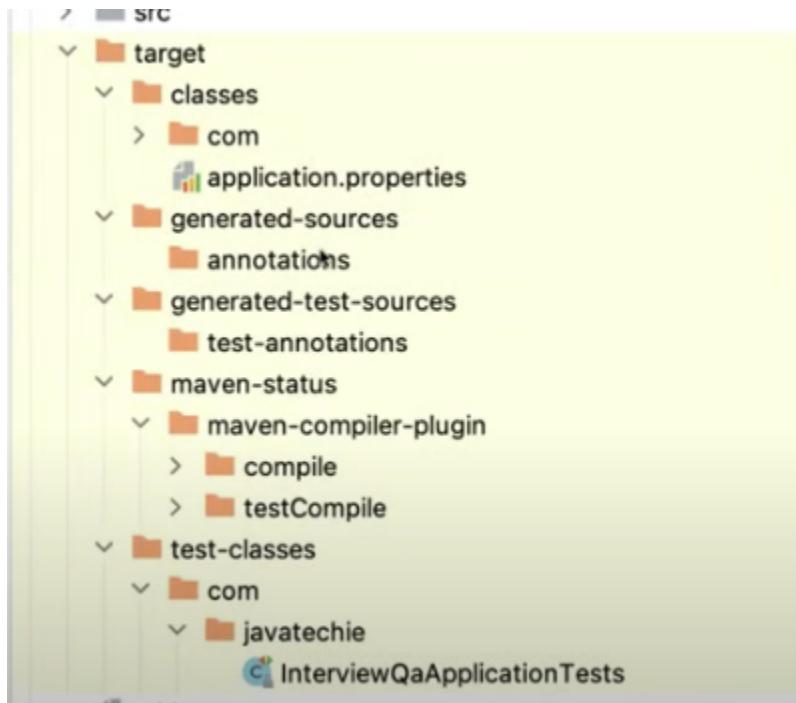
  jpa:
    hibernate:
      ddl-auto: update
      naming:
        physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
      show-sql: true
      defer-datasource-initialization: true

  properties:
    hibernate:
      dialect: org.hibernate.dialect.MySQLDialect
      format_sql : true
```

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.

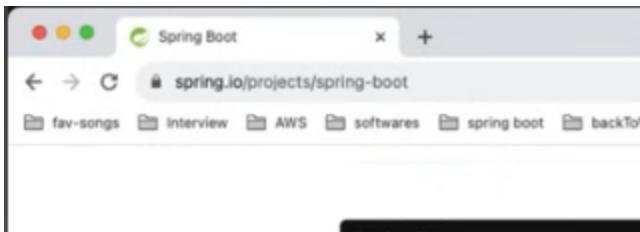


Provide production-ready features such as metrics, health checks

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.

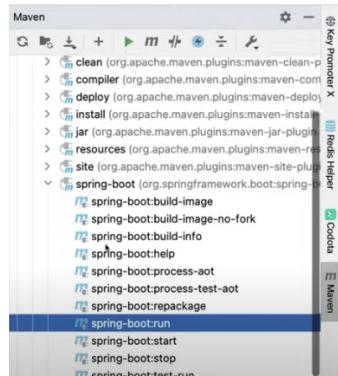
2. What all spring boot starter you have used or what all module you have worked on ?

- 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 ?



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 %
```

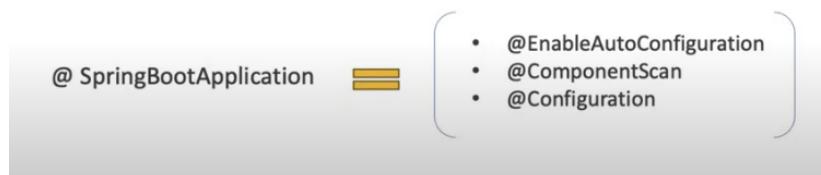


In BOOT-INF it contains classes and lib

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

```
51  * @Target(ElementType.TYPE)
52  * @Retention(RetentionPolicy.RUNTIME)
53  * @Documented
54  * @Inherited
55  * @SpringBootConfiguration
56  * @EnableAutoConfiguration
57  * @ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
58  *                               @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })
59  * public @interface SpringBootApplication {
60 }
```

```
79  * @Documented
80  * @Inherited
81  * @AutoConfigurationPackage
82  * @Import(AutoConfigurationImportSelector.class)
83  * public @interface EnableAutoConfiguration {
84
85
86      Environment property that can be used to override
87      when auto-configuration is enabled.
88
89      String ENABLED_OVERRIDE_PROPERTY = "spring.boot.enableautoconfiguration";
90
91
92      Exclude specific auto-configuration classes such that
93      they will never be applied.
94 }
```

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.

```
52  * @Retention(RetentionPolicy.RUNTIME)
53  * @Documented
54  * @Inherited
55  * @SpringBootConfiguration
56  * @EnableAutoConfiguration
57  * @ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
58  *                               @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })
59  * public @interface SpringBootApplication {
```

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

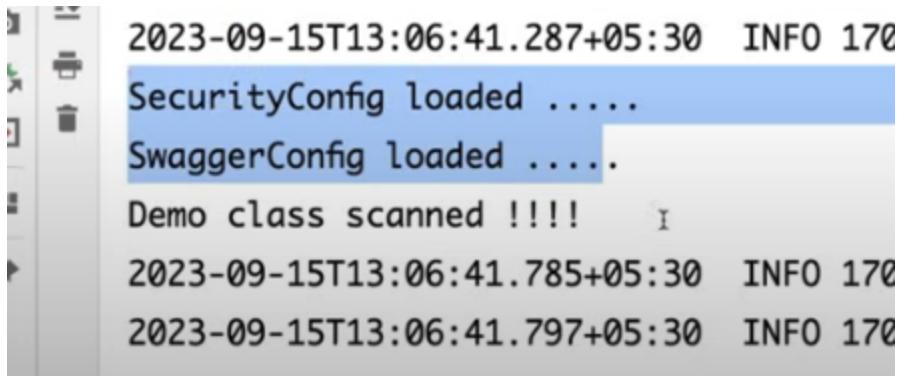
```
1 @SpringBootApplication(scanBasePackages = {"com.business.common", "com.javatechie.*"})  
2 public class InterviewQaApplication {  
3 }
```

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

```
1 @Configuration  
2 public class SwaggerConfig {  
3  
4     public SwaggerConfig() {  
5         System.out.println("SwaggerConfig loaded .....");  
6     }  
7 }
```

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 }
```

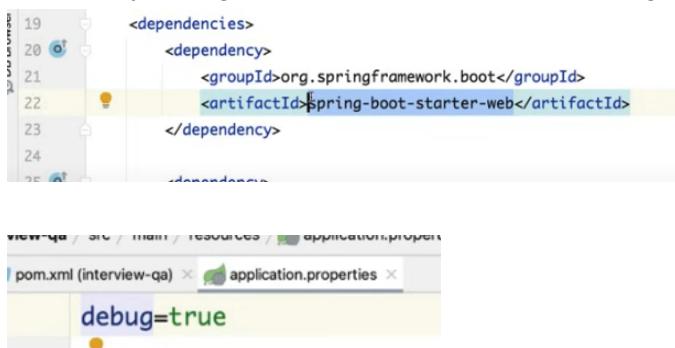


```
2023-09-15T13:06:41.287+05:30  INFO 170 --- SecurityConfig loaded ....  
SwaggerConfig loaded ....  
Demo class scanned !!!!  
2023-09-15T13:06:41.785+05:30  INFO 170  
2023-09-15T13:06:41.797+05:30  INFO 170
```

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



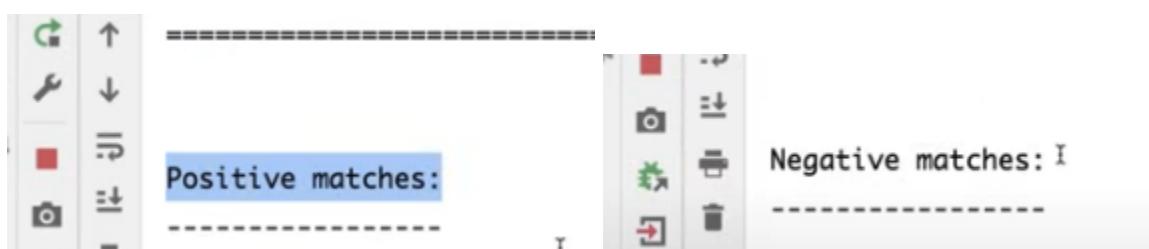
pom.xml (interview-qa) application.properties

```
19  
20     <dependency>  
21         <groupId>org.springframework.boot</groupId>  
22         <artifactId>spring-boot-starter-web</artifactId>  
23     </dependency>  
24 
```

application.properties

```
debug=true
```

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



Positive matches:

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' (OnClassCondition)

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>-->

```

avortes project > dependencies

you will see DataSourceAutoConfiguration in the positive matches

```

Run: InterviewQaApplication
DataSourceAutoConfiguration
ApplicationAvailabilityAutoConfiguration#applicationAvailability matched:
- @ConditionalOnMissingBean (types: org.springframework.boot.availability.ApplicationAvailability; SearchStrategy: all) did not find any beans (OnBeanCondition)

DataSourceAutoConfiguration matched:
- @ConditionalOnClass found required classes 'javax.sql.DataSource', 'org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseType' (OnClassCondition)
- @ConditionalOnMissingBean (types: io.r2dbc.spi.ConnectionFactory; SearchStrategy: all) did not find any beans (OnBeanCondition)

DataSourceAutoConfiguration_PooledDataSourceConfiguration matched:
- AnyNestedCondition 1 matched 1 did not; NestedCondition on DataSourceAutoConfiguration.PooledDataSourceCondition.PooledDataSource found supported
- @ConditionalOnMissingBean (types: javax.sql.DataSource,javax.sql.XADatasource; SearchStrategy: all) did not find any beans (OnBeanCondition)

DataSourceAutoConfiguration_PooledDataSourceConfiguration#jdbcConnectionDetails matched:
- @ConditionalOnMissingBean (types: org.springframework.boot.autoconfigure.jdbc.JdbcConnectionDetails; SearchStrategy: all) did not find any beans (OnBeanCondition)

```

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

```

Shimizu
@AutoConfiguration(before = SqlInitializationAutoConfiguration.class)
@ConditionalOnClass({ DataSource.class, EmbeddedDatabaseType.class })
@ConditionalOnMissingBean(type = "io.r2dbc.spi.ConnectionFactory")
@EnableConfigurationProperties(DataSourceProperties.class)
@Import(DataSourcePoolMetadataProvidersConfiguration.class)
public class DataSourceAutoConfiguration {

    @Configuration(proxyBeanMethods = false)
    @Conditional(EmbeddedDatabaseCondition.class)
    @ConditionalOnMissingBean({ DataSource.class, XADatasource.class })
    @Import(EmbeddedDataSourceConfiguration.class)
    protected static class EmbeddedDatabaseConfiguration {
}

```

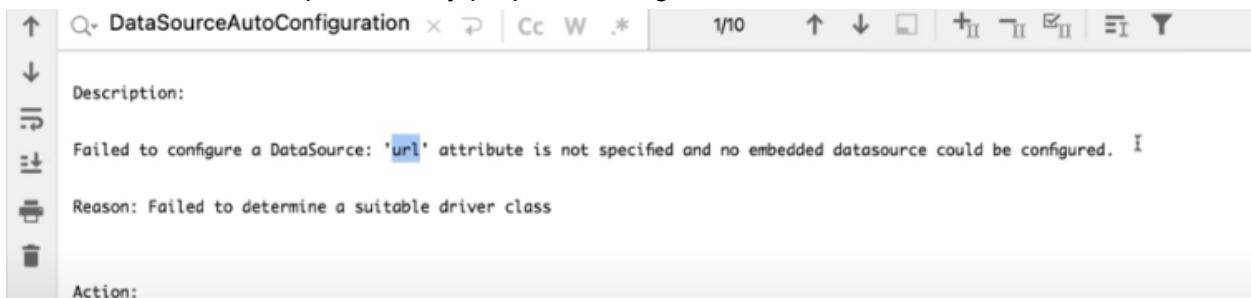
Click on DataSourceProperties.class to see what are all properties it is expecting.

```

47     Benedikt Ritter, Eddú Meléndez, Scott Frederick
48
49
50
51
52
53
54
55
56
57     @ConfigurationProperties(prefix = "spring.datasource")
58
59     public class DataSourceProperties implements BeanClassLoaderAware, InitializingBean {
60
61         private ClassLoader classLoader;
62
63
64
65
66
67     private Class<? extends DataSource> type;
68
69         Fully qualified name of the JDBC driver. Auto-detected based on the URL by default.
70         private String driverClassName;
71
72         JDBC URL of the database.
73         private String url;
74
75         Login username of the database.
76         private String username;
77
78         Login password of the database.
79         private String password;
80
81
82
83
84
85
86
87
88

```

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



Now add these properties and try

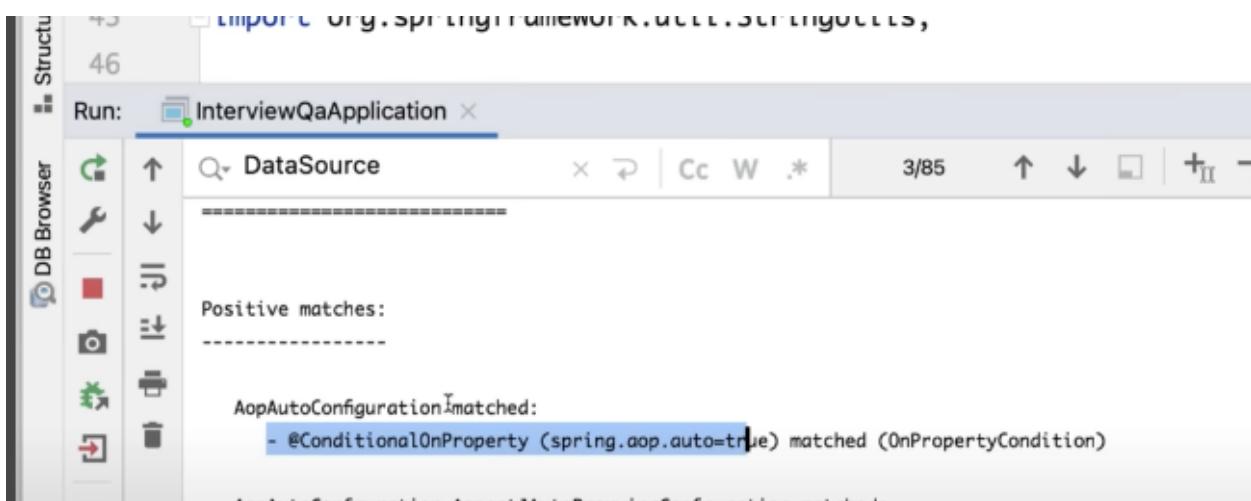
```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/javatechie
spring.datasource.username = root
spring.datasource.password = Password
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamin
```

The screenshot shows the application.properties file in a Java IDE. It contains the following properties:

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/javatechie
spring.datasource.username = root
spring.datasource.password = Password
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamin
```

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.

```

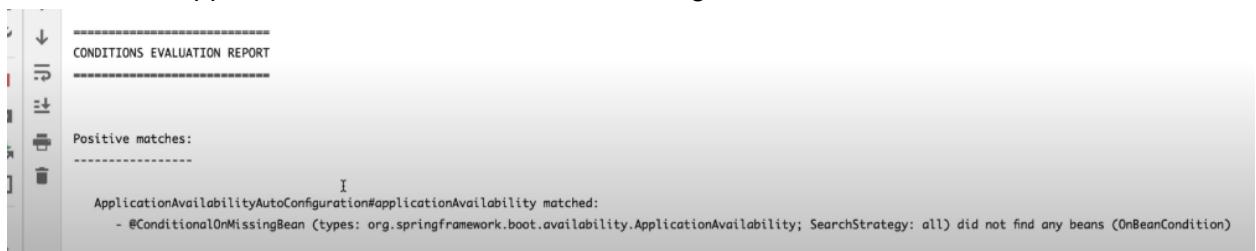
Author: Dave Syer, Josh Long
46     @AutoConfiguration
47     @ConditionalOnProperty(prefix = "spring.aop", name = "auto", havingValue = "true", matchIfMissing = true)
48     public class AopAutoConfiguration {
49
50         @Configuration(proxyBeanMethods = false)
51         @ConditionalOnClass(Advice.class)

```

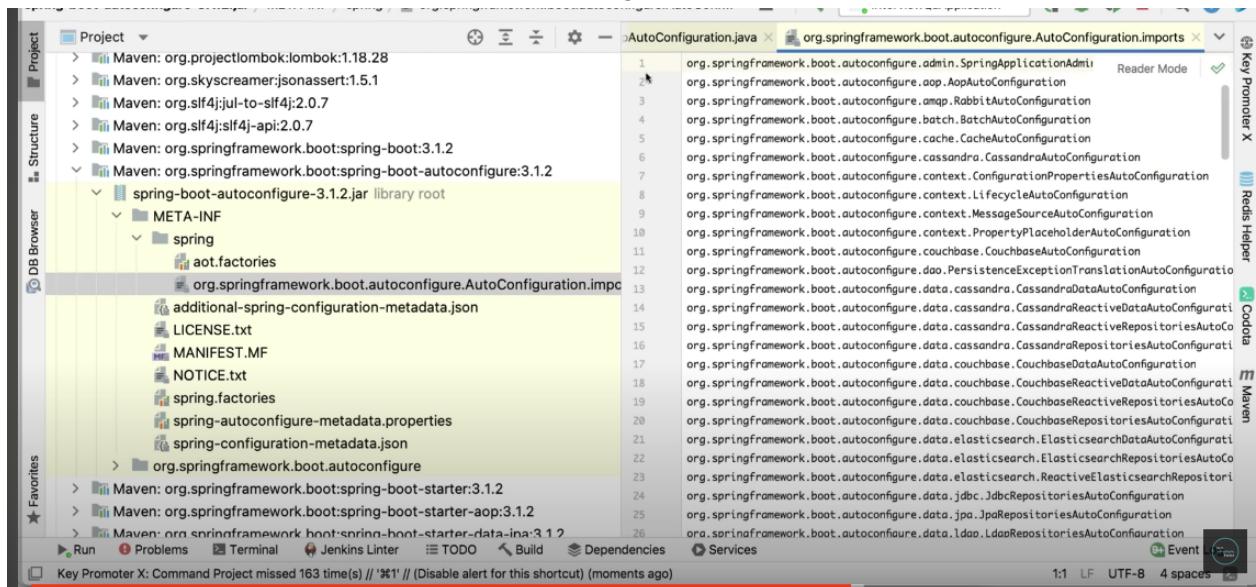
Now override that property



And start the application. Now we won't 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 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 overriding as shown below

```
InterviewQaApplication.java X application.properties  
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.MySQL5InnoDB
```

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

The screenshot shows the IntelliJ IDEA interface during a debug session. The current file is `SpringApplication.java`, specifically the `run` method. Several breakpoints are set along the code, indicated by red dots. The code itself is as follows:

```
passed from a Java main method)
Returns: a running ApplicationContext
public ConfigurableApplicationContext run(String... args) {
    long startTime = System.nanoTime();
    DefaultBootstrapContext bootstrapContext = createBootstrapContext();
    ConfigurableApplicationContext context = null;
    configureHeadlessProperty();
    SpringApplicationRunListeners listeners = getRunListeners(args);
    listeners.starting(bootstrapContext, this.mainApplicationClass);
    try {
        ApplicationArguments applicationArguments = new DefaultApplicationArguments(args);
        ConfigurableEnvironment environment = prepareEnvironment(listeners, bootstrapContext, applicationArguments);
        Banner printedBanner = printBanner(environment);
        context = createApplicationContext();
        context.setApplicationStartup(this.applicationStartup);
        prepareContext(bootstrapContext, context, environment, listeners, applicationArguments, printedBanner);
        refreshContext(context);
        afterRefresh(context, applicationArguments);
        Duration timeTakenToStartup = Duration.ofNanos(System.nanoTime() - startTime);
    }
```

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

The screenshot shows the IntelliJ IDEA interface with the code editor open. Line 9 contains the assignment of the `run` method to a variable named `context`:

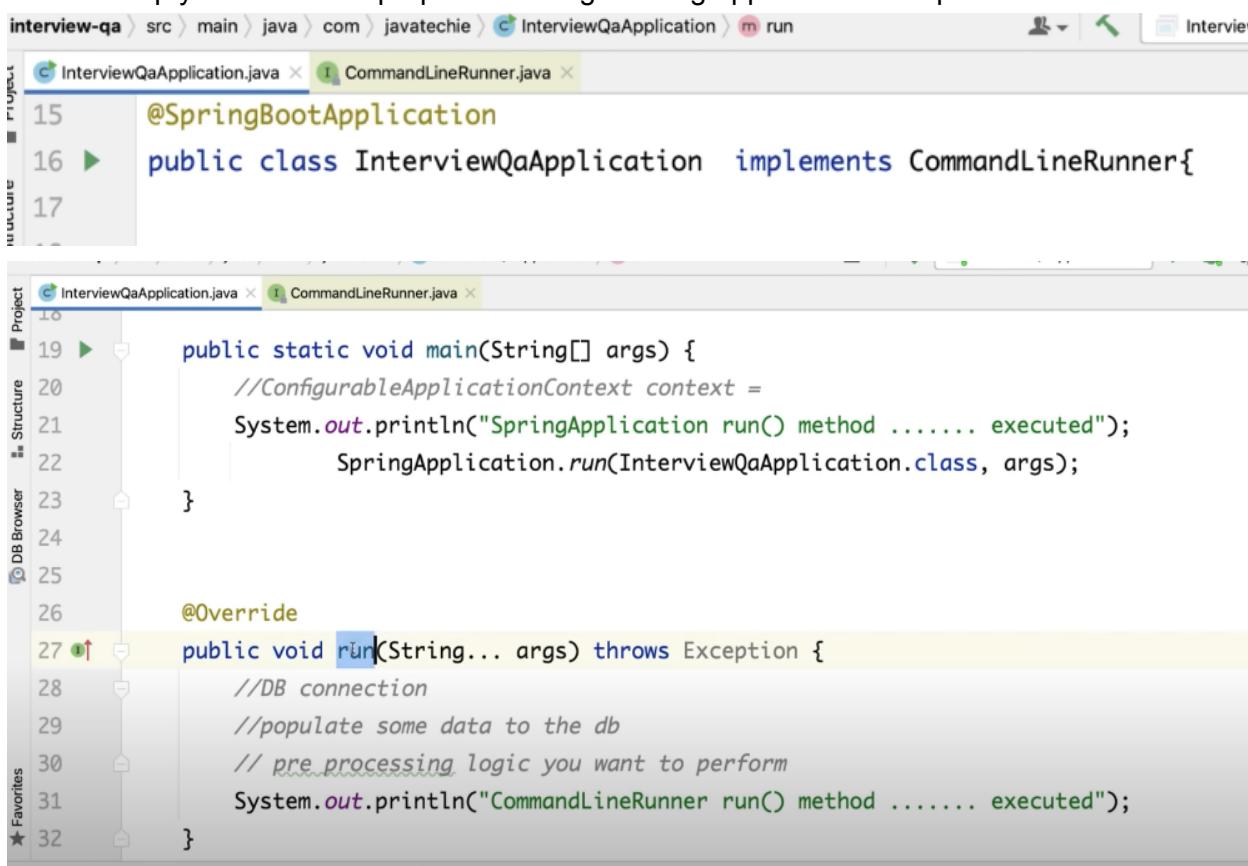
```
8
9 public static void main(String[] args) {
0     ConfigurableApplicationContext context = SpringApplication
1         .run(InterviewQaApplication.class, args);
2 }
```



```
44
45
46    @Override
47    public WebApplicationContext create(WebApplicationType webApplicationType) {    webApplicationType
48        if (webApplicationType != WebApplicationType.SERVLET) ? null : createContext();    webApp
```

10. What is Command line runner in spring boot ?

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



```
15  @SpringBootApplication
16  public class InterviewQaApplication implements CommandLineRunner{
17
18
19  public static void main(String[] args) {
20      //ConfigurableApplicationContext context =
21      System.out.println(".SpringApplication run() method ..... executed");
22      SpringApplication.run(InterviewQaApplication.class, args);
23  }
24
25
26  @Override
27  public void run(String... args) throws Exception {
28      //DB connection
29      //populate some data to the db
30      // pre_processing logic you want to perform
31      System.out.println("CommandLineRunner run() method ..... executed");
32  }
}
```

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.