Applied research

Name: Lucas Jacobs
Pcn: 490692
Class: S3 CB-05
Teachers: Jacco Snoerren, Maja Pesic
Date: 04-11-2022
Word count: 2859

Version History

Version	Date	Author(s)	Changes	State
1	2022-10-07	Lucas Jacobs	Topic, main and sub-questions, method describing and first research answers on first and second sub-question	In progress
2	2022-11-02	Lucas Jacobs	A method added ‘prototyping’ for sub-question 3, added the answers on the sub-questions plus the conclusion, and wrote the references down with APA style	Finished

Contents

Problem description

Within this document, I am going to research an annotation in the framework called Spring Boot. This annotation is called ‘@Autowire’. Spring Boot is a framework that you can use in the code language Java. To do this research, I do not have enough detailed information about the Spring Boot framework in relation to the @Autowire.

Main question

How does @Autowired actually work in Spring Boot?

Sub questions

1. What kind of function can you do with @Autowired that makes it so that people want to use the annotation?
2. Has @Autowired actually benefits compared to the standard way of implementing dependency injection in Java? If so, what?
3. How can you implement the annotation @Autowired in your code?
4. What kind of logic is behind the annotation @Autowired so that it makes it possible to use dependency injection within Java?

Methods to answer the sub-questions

Sub-question 1

For this question, I want to use field research. This is because the question is about exploring the application context. With this question, I will get a more in-depth knowledge of why people are using the annotation @Autowired.
To continue, the first method that I am going to use is problem analysis. This is because you need to know for instance ‘Why is this function satisfied for the user?’. By asking this kind of question you will get a better understanding of the actual answer to the sub-question. Furthermore, I want to use the method called observation. With this method, I will get a more clear view of how people are going to use this annotation. (ictresearchmethods, 2019)

Sub-question 2

With this question, I, first of all, want to use community research. With this research, I can tackle the problem by seeing what other people think of the @Autowired annotation. I can go to online communities like Stack Overflow. By combining the knowledge I have gained from it I can conclude if there will be actual benefits.
To continue, I also want to use a literature study for this question. I first need to find general information before I can conclude for myself if @Autowired has some benefits. (ictresearchmethods, 2019)

Sub-question 3

To this question, I want to apply the method ‘Best good and bad practices. This is mainly because I want to figure out how other people have implemented the annotation @autowire. So to conclude, this is also better to understand if some people struggled to implement @autowire, or if the implementation is fairly simple. Finally, I want to use the ‘prototyping method’ so I can implement it by myself and have a better understanding of how I can use it. (ictresearchmethods, 2019)

Sub-question 4

To begin with the first method, I will use the ‘Literature study’ again. The biggest reason for this is that I need get in-depth information about the annotation. When I am done with this method and I have written a summary of my findings then I can continue with the method of ‘Problem analysis. This way I will get an understanding of the actual task that @autowire is performing. (ictresearchmethods, 2019)

Answers sub-questions

Sub-question 1

Problem analysis

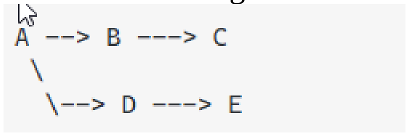
First of all, let's start by asking what can be accomplished by using @autowired. @autowired is a feature of the Spring Boot framework. The annotation will be used to enable us to inject the dependency object naturally (Educba,). In other words, @autowired will say ‘can you give me an instance of this class?’. Compared to the @Bean which you will use to keep hold of an instance of a class and you can ask for it when to pass it back. (Stack Overflow, 2015)
To continue, where can @autowire be used? There are multiple ways of implementing autowiring in your code. The following are:

- Nothing: this is the default autowiring mode. It just means that there is no autowiring.
- byName: In this mode ‘byName’ inject the object dependency by the name of the bean. In this case, the property and bean name needs to be the same.
- byType: this is the same as a name but it injects the object dependency according to type. This way, it can have a different bean and property name.
- Constructor: Autowiring with a constructor is similar to ‘byType’ but then it is for constructors’ arguments.
- Autwiring with autodetect: this will automatically try to autowire by the constructor, but when it fails it will try to autwire it byType.

(DZone)

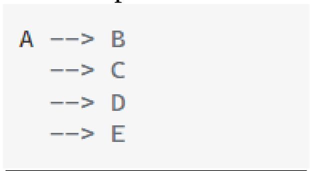
Observation

Next, an important thing to ask is, why is this function @autowired satisfied for the users? On StackOverflow, I found something under the question: Why do we use @Autowire? Here I found that @autowire avoids that you have complex tree structures between your classes. See the figure below for an example of a complex structure and a solution to it.



```
A --> B ---> C
 \
  \--> D ---> E
```

Figure 1: classes complex tree structure



```
A --> B
  --> C
  --> D
  --> E
```

Figure 2: solution

So in the context of @autowire, You declare for spring what class you want to be passed in the instantiated classes. When Spring starts, it will look at which class is fitted where. (StackOverflow, 2020) This way people want to use @autowire because it is good for structuring the code.
Moreover, I found something else on StackOverflow which said two main things for using @autowire. First of all, when you use @autowire it will reduce the need to specify constructor or properties arguments. Secondly, auto wiring will update your configuration automatically when your objects evolve. When you add a dependency to a class it will automatically update it in the configuration. This is quite beneficial because you will spare time writing code and you do not have to think about it manually configuring each class. (StackOverflow, 2019)

Conclusion

People can use @autowire for multiple reasons. To begin with, when you want to use the function to work at its finest you need to apply it in your project consistently. The main function of @autowire is automatically applying dependency injection. The benefits of @autowire are that it automatically updates your configurations of classes and it reduces writing code due to you don’t have to instantiate properties and constructors by yourself.

Sub-question 2

Literature study

To begin with, it is beneficial for writing less code. When you are using the annotation, you will write less code for the reason that you do not need to write code to inject the dependency explicitly. Therefore people will use this framework because it will reduce developing time since you do not need to specify the properties and constructor arguments by yourself. (DZone,)
On the other hand, there are also some disadvantages of autowiring. Autowiring is less precise than explicit wiring. Spring tries to avoid double meaning by not explicitly stating the relationships between objects. Also, when you want to document your code, tools will maybe not support a spring container.
Autowiring in Spring Boot also has its limitations to it. These are the following:

- When there are a lot of dependencies in a program, it is hard to find them using the autowire attribute of a bean.
- You cannot autowire primitive data types and strings. It only works with references.
- Overriding is possible. It will always override when you define a dependency using a property or constructor-args tag.

(DZone)

Community Research

On the website ‘A Java geek’, A developer with 15 plus years of experience the day that he wrote the blog, wrote a blog about why he is against auto wiring. Auto wiring is there to simplify dependency injection. Auto wiring in spring is approximately expressing constraints on all accessible beans within the spring setting. This way one and only one matches the required dependency. When you wire to spring, you entrust him to discover the correct bean for you. The downside is that you have the chance that you run in that it finds more than one match. So in real life when we have a large project, where we have multiple teams working, we will have a higher risk to have multiple matches which leads to bugs in the code. (A Java geek, 2013)

Conclusion

When you have a project for yourself it is fine to use auto wiring. It for example reduces developing time because it automatically configures your dependencies. When you have a large team working on a big project, auto-wiring can lead to bugs and it can be confusing what dependencies there are. Therefore in a group, it is better to have manual configurations for dependencies.

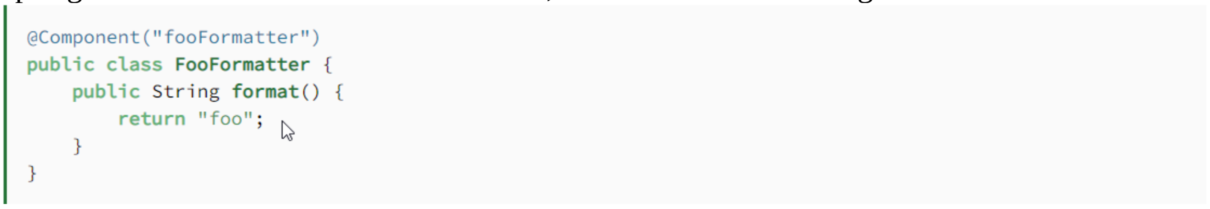
Sub-question 3

Best good and bad practices

@Autowired can be used in multiple ways, for example on properties, setters and constructors.

Property

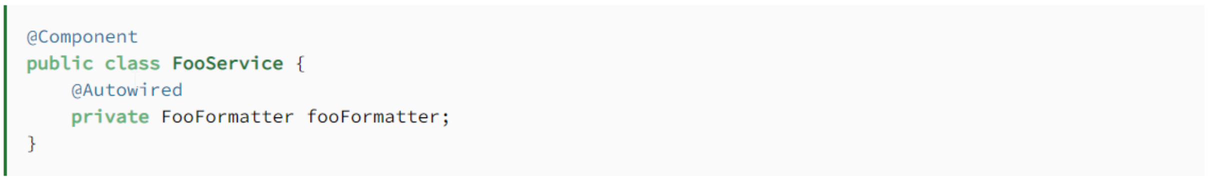
Let have a look how it can be implemented on a property. First of all make a object or in spring a bean with the name FooFormatter, this will look something like this:



```
@Component("fooFormatter")
public class FooFormatter {
    public String format() {
        return "foo";
    }
}
```

Figure 3: FooFormatter bean

Now we want to inject this object (bean) to the FooService by using the @Autowired definition.

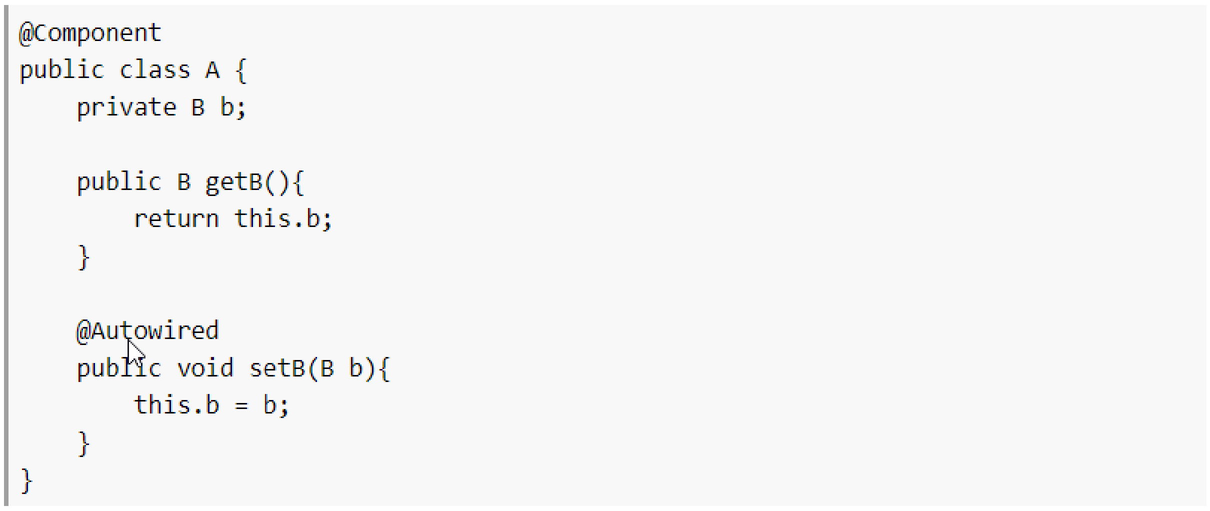


```
@Component
public class FooService {
    @Autowired
    private FooFormatter fooFormatter;
}
```

Figure 4: FooService

When FooService will be created then Spring injects fooFormatter.(Baeldung, 2022)

Setter



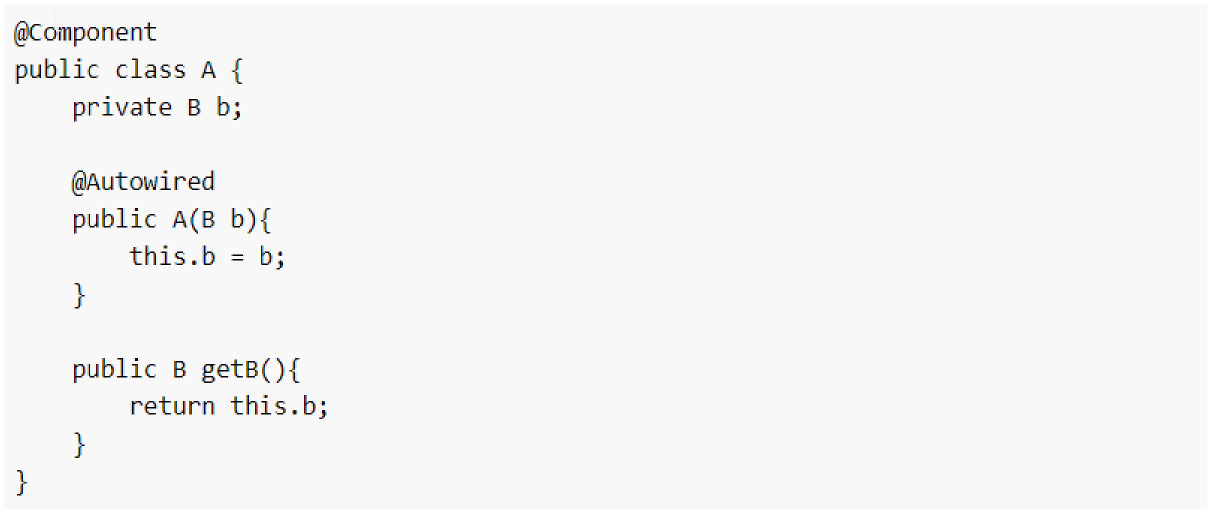
```
@Component
public class A {
    private B b;

    public B getB(){
        return this.b;
    }

    @Autowired
    public void setB(B b){
        this.b = b;
    }
}
```

Figure 5: @Autowired in setter method

This is really useful because: “Annotating setter methods with @Autowired removes the need to explicitly set B when initializing A”. (stackchief)

When something is annotated with @Autowired, when A will be initialised then the setter method will be called automatically. It will then pass the Spring managed B as an argument. (stackchief)

Constructor



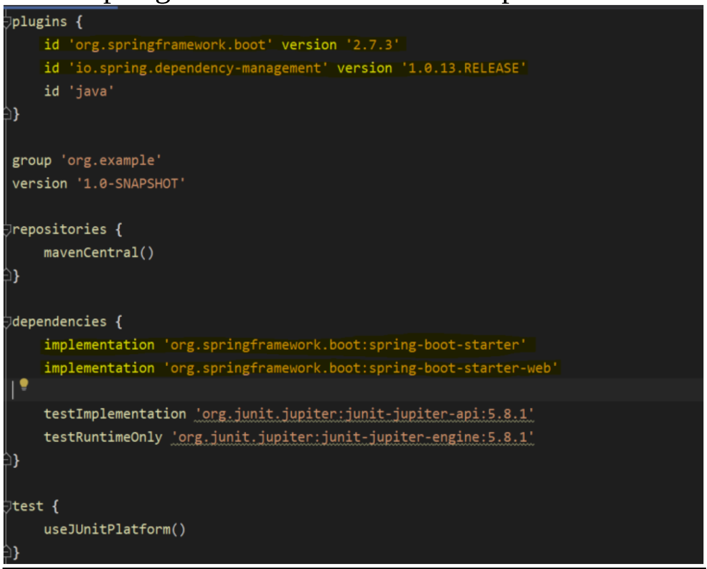
```
@Component
public class A {
    private B b;

    @Autowired
    public A(B b){
        this.b = b;
    }

    public B getB(){
        return this.b;
    }
}
```

Figure 6: @Autowired used on a constructor

When we put the annotations above the constructor, automatically inject Spring managed B when B is initialized. This is can be really useful because “managed dependencies are explicitly required for creating A”(stackchief). With this it will make testing al lot simpler because this ensure that you do not have any runtime exceptions when for in this example B is not managed or instantiated.
With newer spring version starting from 4.3 and above, you do not longer have to write @Autowired on constructors. When you constructor only has one bean argument, Spring will only then automatically inject the managed dependency. (stackchief)

prototyping method

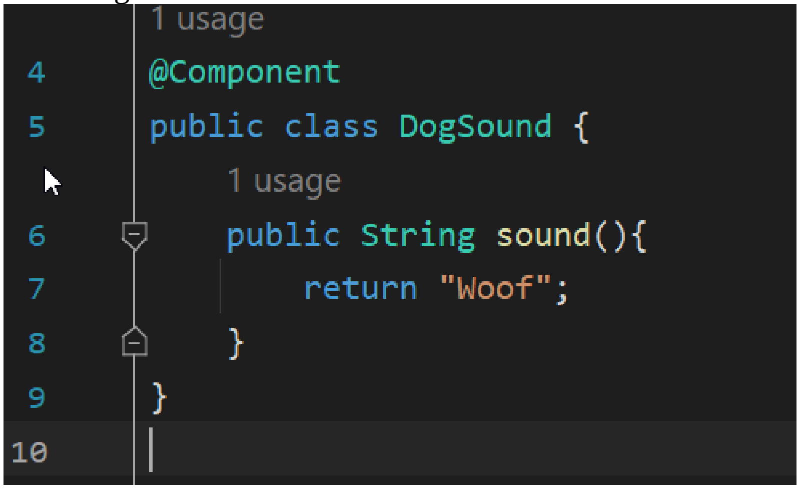
Let’s first start by creating a new project in IntelliJ, which is called “TestAutowired”. When it is created we first want to make some adjustments in ‘build.gradle’ some we can make use of the Spring boot framework. See the picture what needs to be added.



```
plugins {
    id 'org.springframework.boot' version '2.7.3'
    id 'io.spring.dependency-management' version '1.0.13.RELEASE'
    id 'java'
}

group 'org.example'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    implementation 'org.springframework.boot:spring-boot-starter-web'

    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'
}

test {
    useJUnitPlatform()
}
```

Figure 7: dependencies for Spring

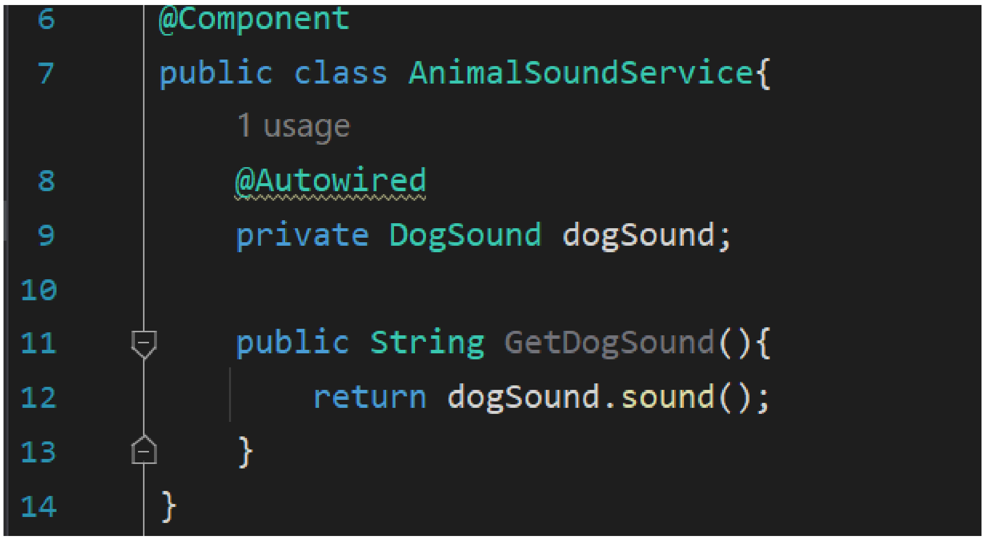
Next up, I will create a class called “DogSound” which will return a String with the sound that a dog makes.



```
1 usage
@Component
public class DogSound {
    1 usage
    public String sound(){
        return "Woof";
    }
}
```

Figure 8: DogSound class

Furthermore we have a class called ‘AnimalSoundService”.



```java
@Component
public class AnimalSoundService{
    1 usage
    @Autowired
    private DogSound dogSound;

    public String GetDogSound(){
        return dogSound.sound();
    }
}
```

Figure 9: Service class

When AnimalSoundService is created then dogSound will be injected and ready to use. Moreover, the method GetDogSound can then return the dog sound of the Object DogSound.

Sub-question 4

Literature study

To instantiate a @autowired class, Java uses java reflection. First, what is java reflection? It allows an executing java program to examine or “introspect” upon itself, and manipulate the internal properties of the program. For example, a Java class can obtain the names of all its members and display them. (Oracle, 1998)
Java reflection is an API, to give a visual illustration of what the API does (geeksforgeeks, 2022):

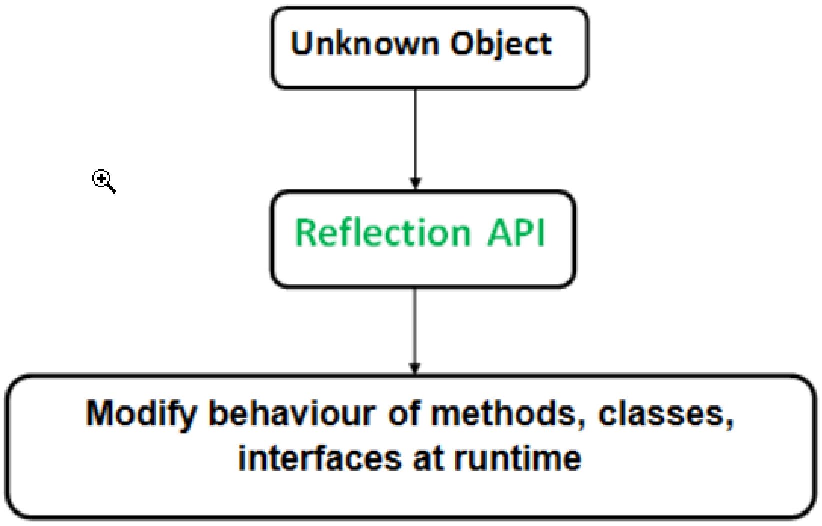


Figure 10: reflection API example

When using @Autwired it provides more fine-grained control over where and how auto wiring should be accomplished. (tutorial point)
To continue, what is auto wiring? With Spring we use a so-called ‘Spring Bean Autowiring’, which says that a bean (object) can be declared by Java configurations. By declaring beans you provide metadata(details of configuration properties) send to a Spring container which will return the required dependency object at runtime. In Java-based configuration, all bean methods are defined in classes using the @configuration annotation. At runtime, Spring will read these methods and provide the bean definitions. @Autowired assigns the correct dependencies to the Spring container. (geeksforgeeks, 2022)

Problem analysis

@autowired is a annotation that is not originally from Java but a custom annotation created by Spring. This annotation mainly uses to inject beans or objects into other beans or objects. How does @autowire actually accomplish this? When a Spring application starts, the spring container will make all the objects that are needed. Eventually, one object needs another object, therefore at runtime spring will create a dependent object and injects them. (youtube, 2020)
This is still a bit vague so let’s give an example of how this internally works. To begin in spring we call an object a bean.
In Spring when we put @component on a class, this will create an object for that class. This will be stored in the Spring container, which holds all objects of the application.(medium, 2020)

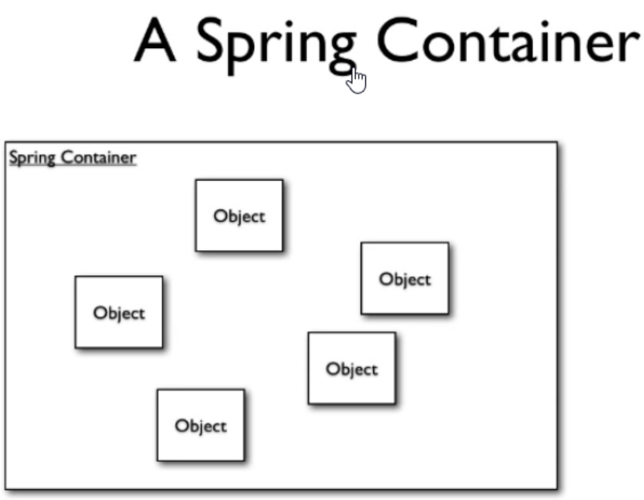
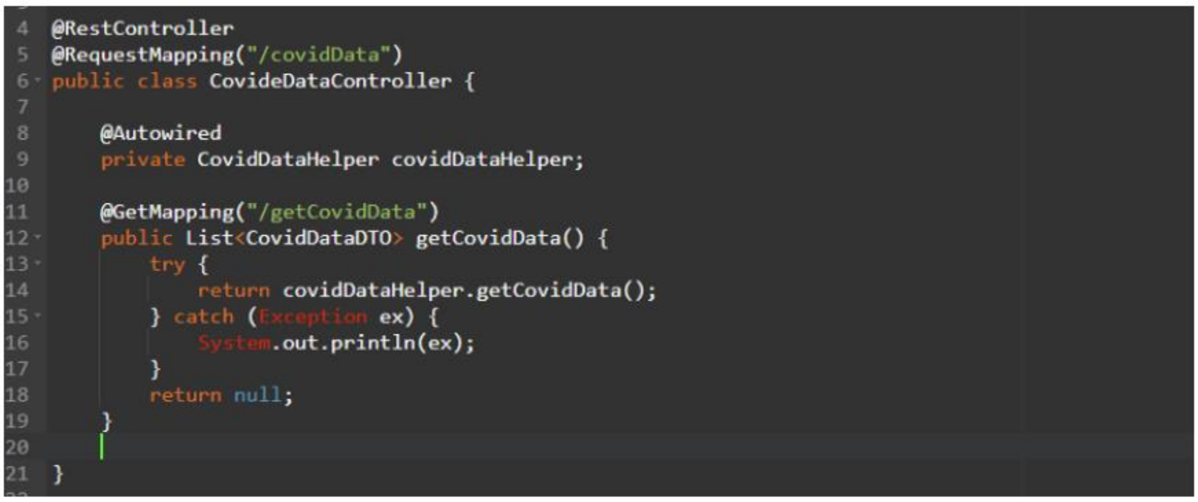


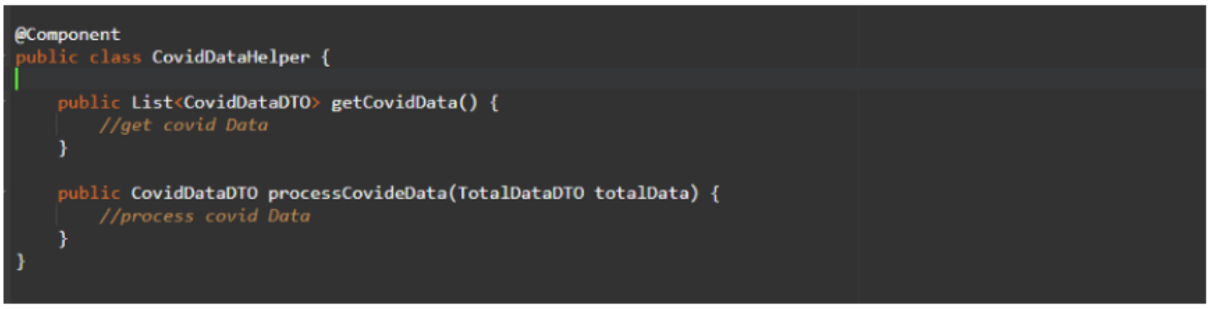
Figure 11: Example of a spring container

See down below for some example code from the site medium.



```
@RestController
@RequestMapping("/covidData")
public class CovideDataController {

    @Autowired
    private CovidDataHelper covidDataHelper;

    @GetMapping("/getCovidData")
    public List<CovidDataDTO> getCovidData() {
        try {
            return covidDataHelper.getCovidData();
        } catch (Exception ex) {
            System.out.println(ex);
        }
        return null;
    }

}
```

Figure 12: controller class



```
@Component
public class CovidDataHelper {

    public List<CovidDataDTO> getCovidData() {
        //get covid Data
    }

    public CovidDataDTO processCovideData(TotalDataDTO totalData) {
        //process covid Data
    }
}
```

Figure 13: helper class

So in this example, writing a @component to a helper class while the application is in run mode will create a bean for the helper class inside the Spring container. The controller class has helper class dependencies. So how does the controller class bean know that there are helper class beans in the container? We use @Autowired when we write @Autowired to a variable in the type helper class because we need to tell the controller class bean to find the helper class bean and use it. It will start looking for a bean in the helper class and inject it into this variable so it is initialized and ready for use. (medium, 2020)

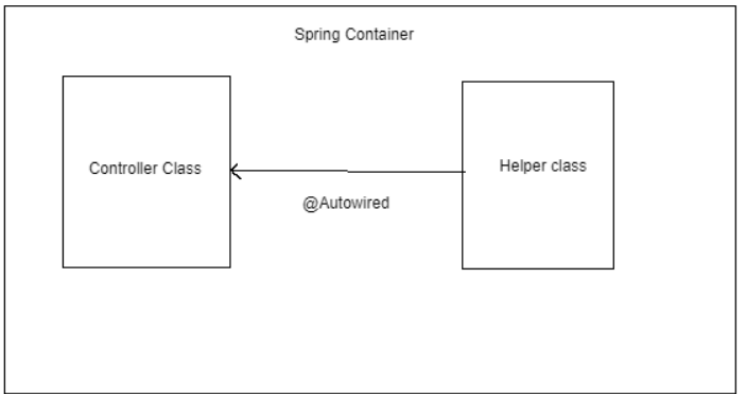


Figure 14: clarifying what happens in the container

Conclusion

@Autowired can be placed above a variable. When the variable uses the @Autowired, then the annotation will look in the Spring container and will try to find the object which then will inject it into the variable, using Java reflection.

Conclusion

The main question was:
How does @Autowired actually work in Spring Boot?

@Autowired is part of Spring which is used in Java and can be used in multiple ways such as for classes, methods, and constructors. To use @Autowired you need to put the annotation above a variable.

What @Autowired does is, when your application runs (runtime), it will search in you Spring container which contains all objects, for the object that belongs to the variable. Furthermore, it will then inject that object into the variable. Finally, you can use the variable with the injection of the object that has been injected into the variable. @Autowired will make use of reflection which makes it possible to give us information about classes.

Benefit of using @Autowired is that you spare development time which can be beneficial for an individual project. When you have multiple teams working on a project, auto-wiring can lead to bugs and it can be confusing what dependencies there are. Therefore in a group, it is better to have manual configurations for dependencies.

Writing this research was pretty difficult, due to the fact that there was not a lot of information available. I think a reason for this has to do with that @Autowired has been deprecated in the newer versions of the Spring framework.

References

@Autowired In Spring Boot. (2020, 05 30). Retrieved from medium: https://medium.com/javarevisited/autowired-in-spring-boot-58fc8a598449

ACADEMY, J. (2020). *Java Annotations | Spring Annotation | @Autowired*. Retrieved from Youtube: https://www.youtube.com/watch?v=Do3l0d5T1cs

Autowiring in Spring. (n.d.). Retrieved from Dzone: https://dzone.com/articles/explain-major-benefits-amp-limits-of-auto-wiring-i

Difference between @Bean and @Autowired. (2015, 12 10). Retrieved from StackOverflow: https://stackoverflow.com/questions/34172888/difference-between-bean-and-autowired

Dzone. (n.d.). Retrieved from Major Benefits and Limits of Autowiring in Spring Java Web development: https://www.educba.com/spring-boot-autowired/

Frankel, N. (2013, 11 03). *My case against autowiring*. Retrieved from A Java geek: https://blog.frankel.ch/my-case-against-autowiring/

Guide to Spring @Autowired. (2022, 09 29). Retrieved from Baeldung: https://www.baeldung.com/spring-autowire

Methods. (2021, 06 08). Retrieved from ictresearchmethods: https://ictresearchmethods.nl/Methods

Reflection in Java. (2022, 05 31). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/reflection-in-java/

Spring @Autowired. (2022, 07 05). Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/spring-autowired-annotation/#:~:text=Spring%20beans%20can%20be%20declared,is%20called%20Spring%20Bean%20Autowiring.

Spring @Autowired Annotation. (n.d.). Retrieved from tutorialpoint: https://www.tutorialspoint.com/spring/spring_autowired_annotation.htm

spring boot autowired. (n.d.). Retrieved from Educba: https://www.educba.com/spring-boot-autowired/

The 5 Minute Guide to Autowire in Spring. (n.d.). Retrieved from stackChief: https://www.stackchief.com/blog/The%205%20Minute%20Guide%20to%20Autowire%20in%20Spring

The DOT Framework. (2021, 6 8). Retrieved from ictresearchmethods: https://ictresearchmethods.nl/The_DOT_Framework

Using Java Reflection. (1998, 01). Retrieved from Oracle: https://www.oracle.com/technical-resources/articles/java/javareflection.html#:~:text=Reflection%20is%20a%20feature%20in,its%20members%20and%20display%20them.

What is the advantage of autowiring in spring. (2019, 11 18). Retrieved from StackOverflow: https://stackoverflow.com/questions/16344062/what-is-the-advantage-of-autowiring-in-spring

Why do we use @Autowire. (2022, 06 07). Retrieved from StackOverflow: https://stackoverflow.com/questions/62249715/why-do-we-use-autowire