A framework is like a structure that provides a base for the application development process. With the help of a framework, you can avoid writing everything from scratch. Frameworks provide a set of tools and elements that help in the speedy development process. It acts like a template that can be used and even modified to meet the project requirements.

Writing the code from scratch is a tedious task full of possible risks and errors. You need to make the code clean, well-tested, bugs and errors free. It will be difficult for other developers to understand the code and work on it. So, it is better to work with the frameworks that meet your requirements. They make the development process easy with fewer errors. It is a general template that can be used and modified as per the requirement. It will be easy for others to understand your code as they are also familiar with frameworks.

Frameworks provide many advantages such as:

- Easy to test your code and debug it.
- Clean code is much easy to understand and work with.
- Reduces redundancy of code in the project.
- Reduces the time and cost of the project with the enhanced application.
- Features and functionalities provided by the framework can be modified and extended.

Spring works with plain old Java Objects.

Spring is used to build Enterprise Applications.

Spring is more famous because of the below

1.Feauters

2.Community support

3.Good Documentation

IOC and DI :

IOC -Inversion of Control(Principle)

DI-Dependency Injection(Design Pattern)

As Developers should be more focussed on writing business logic than the Object creation.

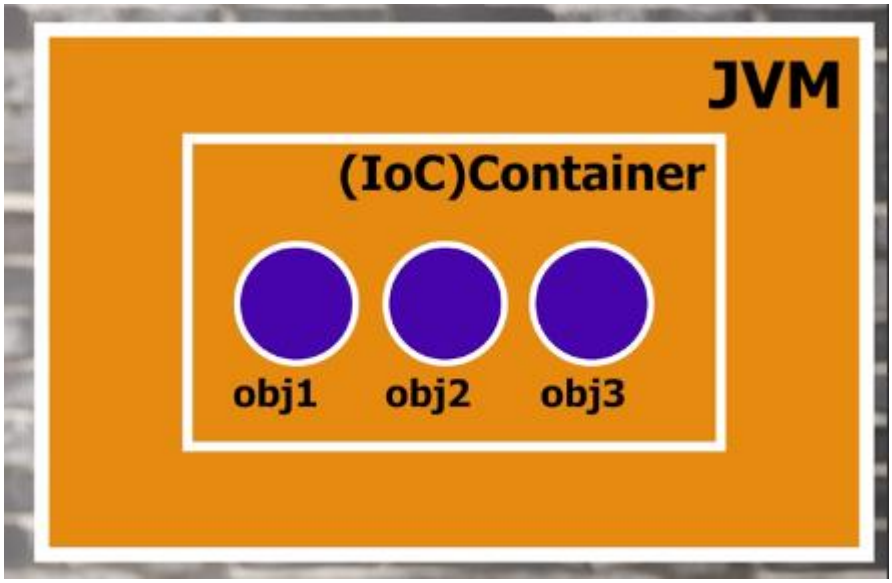
Object creation all these will be taken care Spring Framework.

Spring Framework creates the Objects and will be kept in the IOC Container.

Basically Object creation and control flow will be done by Spring Framework.

IOC Container

Objects will be created by Spring Framework and will be kept in IOC Container.



DI: Dependency Injection(Design Pattern)

Consider we have a Laptop Class and CPU Class.

We need a CPU Object inside laptop class which means laptop Class is dependent on the CPU.

So the CPU Object has to be injected into Laptop Class.

```
class Laptop{
CPU obj;
}

class CPU
{

}
```

Both the Laptop Object and CPU Object will be present in IOC Container, but Laptop class should have CPU Object which we can do using Dependency injection.

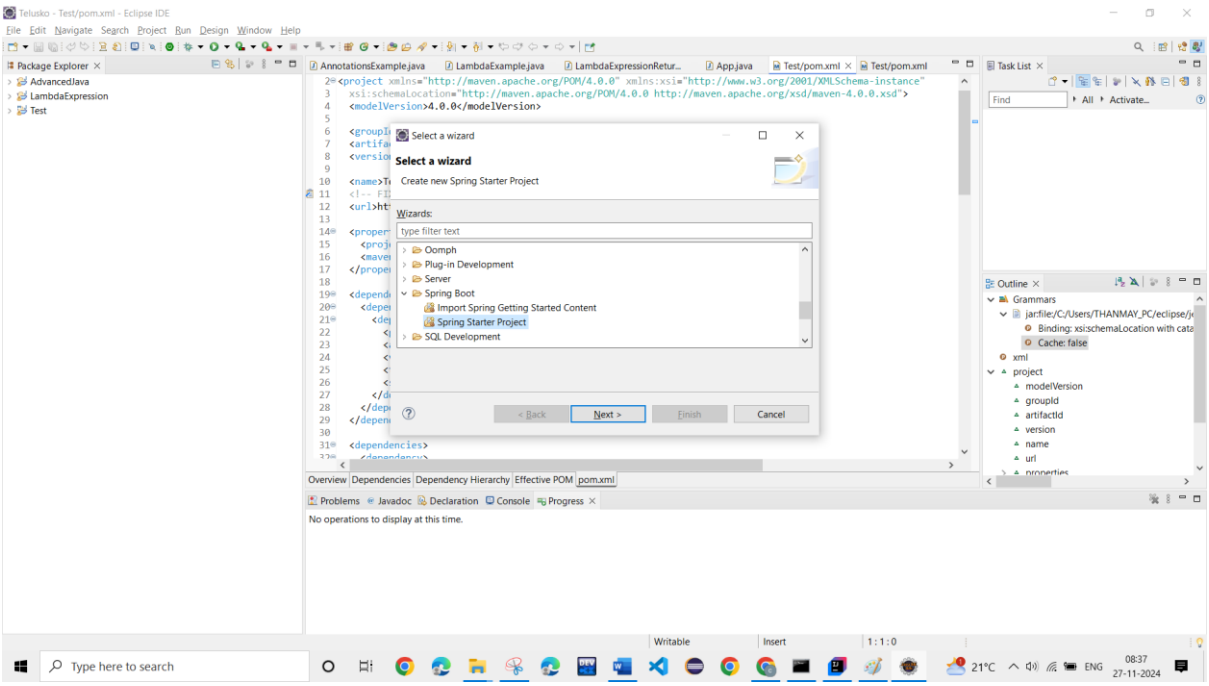
Spring vs Spring Boot:

Spring Boot came on top of Spring .

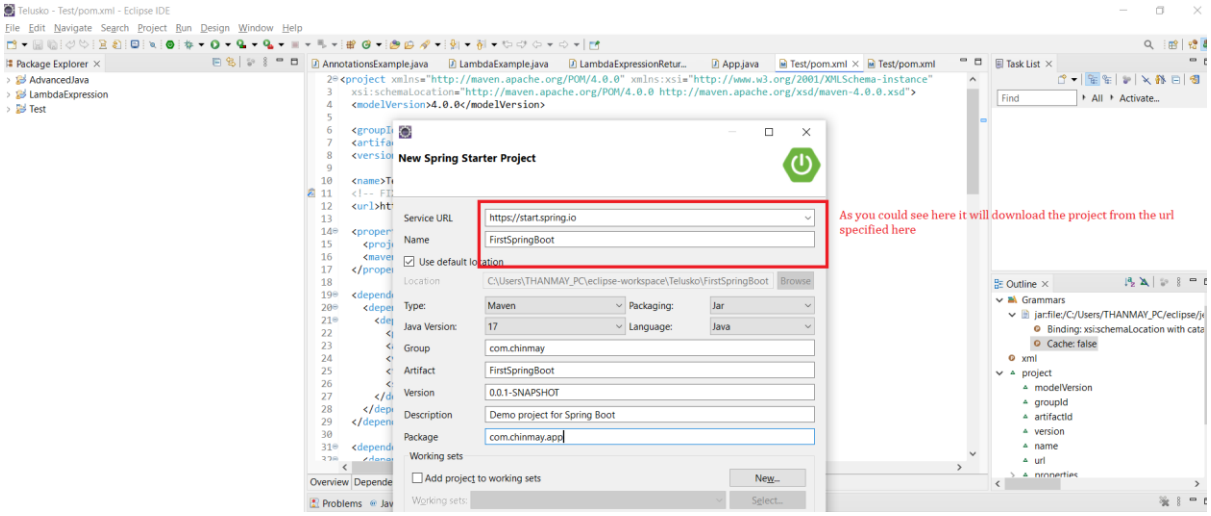
In Spring for running simple Java Applications as well we need to write many configurations to make our application work.

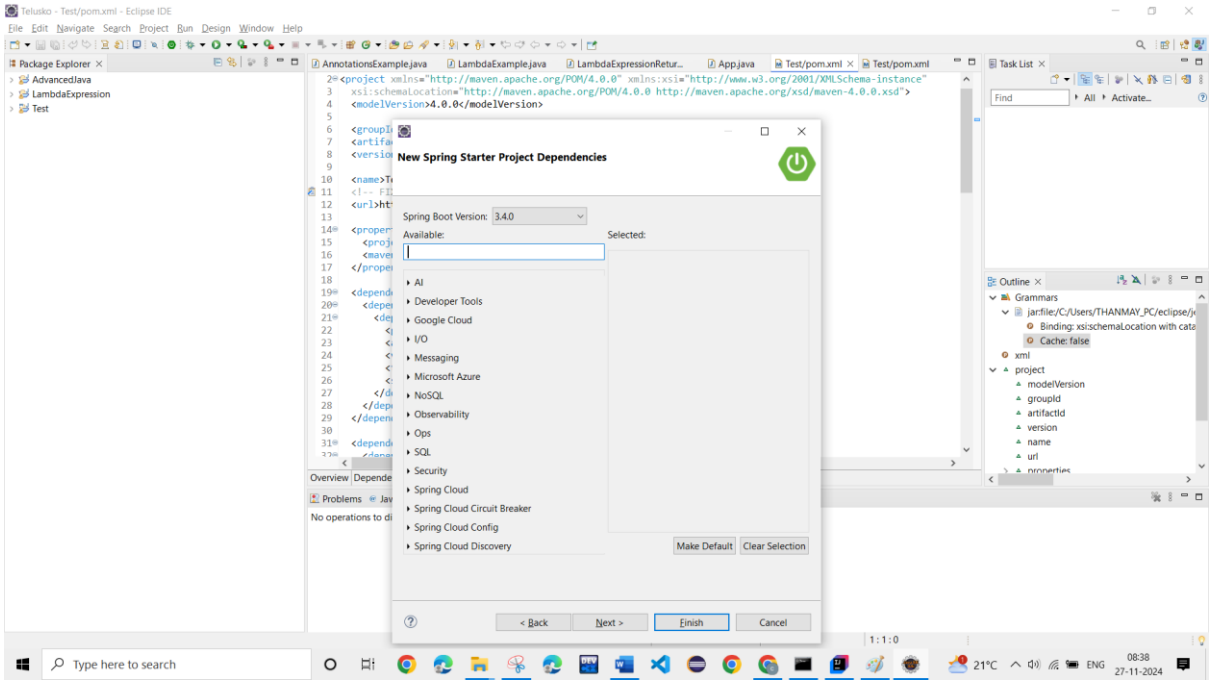
In Case of Spring Boot which came later it will much easier to run our Application as the configuration will be taken care by Spring Boot.

Spring Boot 3 works on Spring 6

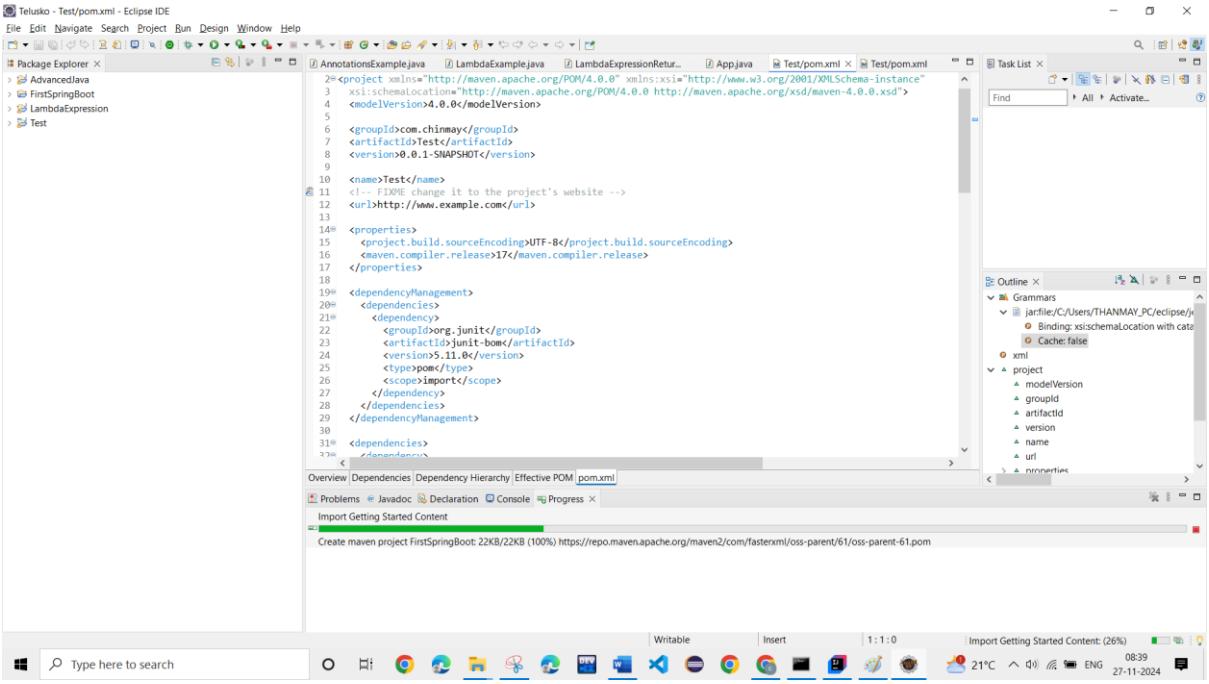


We can select Spring Boot starter Project





If we wan to add any dependencies we can add it here

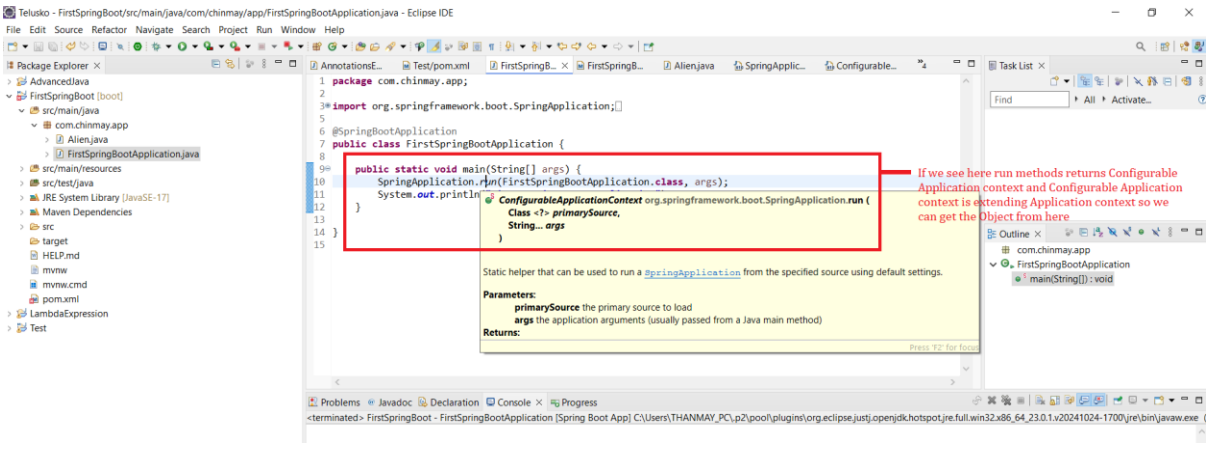


Once we Complete all the above then it will download Spring Boot Starter Project.

Bean: Any Object which is created, managed by Spring will be called beans

Whenever Spring Boot creates an Object it will be present in IOC Container.

So In order to get the Objects from IOC Container we need Application context.

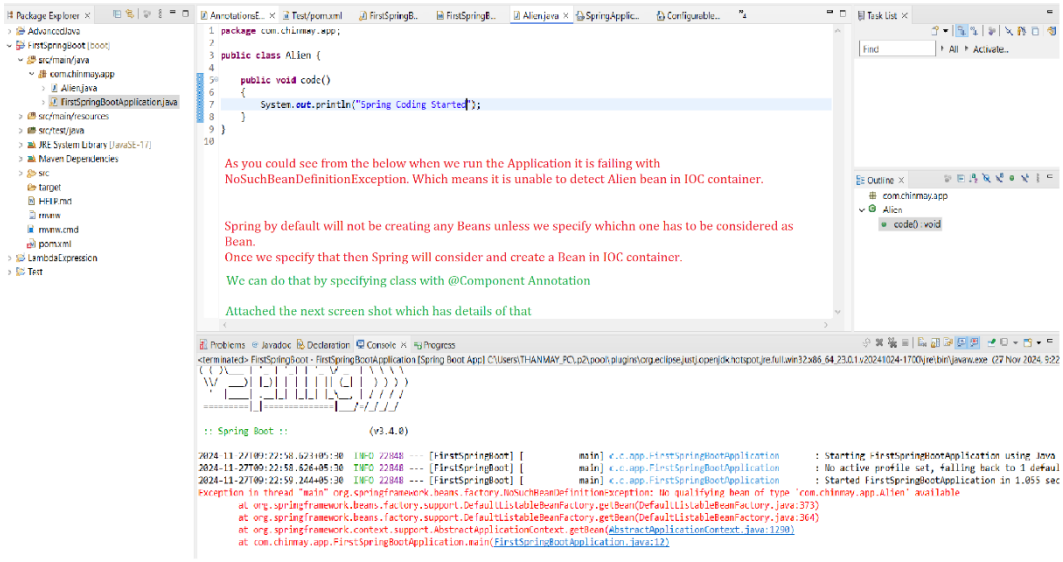



```
46 public interface ConfigurableApplicationContext extends ApplicationContext, Lifecycle, Closeable {
47
48     /**
49      * Any number of these characters are considered delimiters between
50      * multiple context config paths in a single String value.
51      * @see org.springframework.context.support.AbstractXmlApplicationContext#setConfigLocation
52      * @see org.springframework.web.context.ContextLoader#CONFIG_LOCATION_PARAM
53      * @see org.springframework.web.servlet.FrameworkServlet#setContextConfigLocation
54      */
```

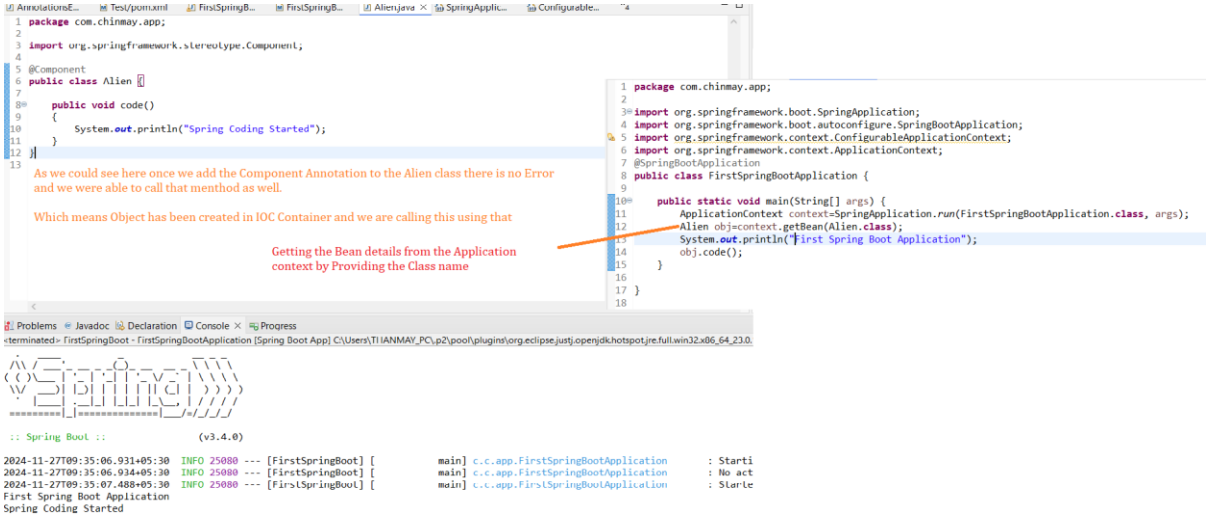
```java
public class FirstSpringBootApplication {

    public static void main(String[] args) {
        ApplicationContext obj1=SpringApplication.run(FirstSpringBootApplication.class, args);
        obj1.getBean(Alien.class);
        System.out.println("First Spring Boot Application");
    }
}
```

Here as the run Returns Application context we are taking that Object and using that Object we are trying to get the Bean Details of Alien class



Resolution for the Above Issue



@Component Annotation:

- @Component is a generic stereotype annotation that can be applied to any class that you want Spring to manage. It's a way to tell Spring that this class is a component of your application and should be managed by the Spring container.
- **Functionality:**

When a class is annotated with @Component, Spring will automatically detect it and create a bean for it in the application context. This allows you to inject the bean into other components using @Autowired or other dependency injection mechanisms.

138.Autowiring in Spring Boot:

Consider a scenario where our class is dependent on Other class and we don't have direct IoC Container then

In the below Scenario Alien class is dependent on Laptop class but the Alien does not have direct access to the container to get the Object/Bean. So in that case we can use @Autowired annotation to the get the bean from the container.