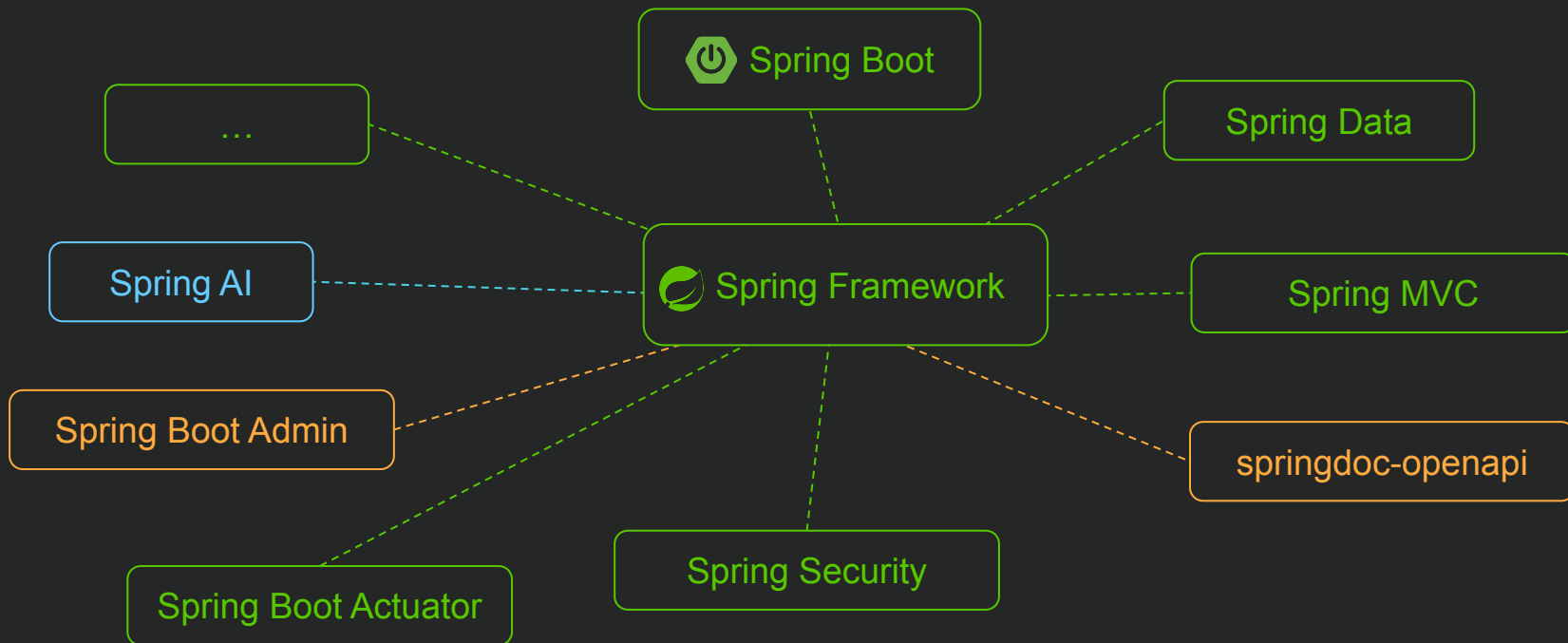


The Spring ecosystem



In green: projects you will use this week and that are maintained by the Spring team at Broadcom

In orange: projects maintained by the community

In blue: projects not taught in the Spring-Core course, taught in a subsequent course

Automated Testing

- In all Labs, you will be using JUnit for Unit Tests
 - Assertions are based on assertJ

```
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.*;

public class MultiplierTest {
    private int multiply(int a, int b) { return a * b; }

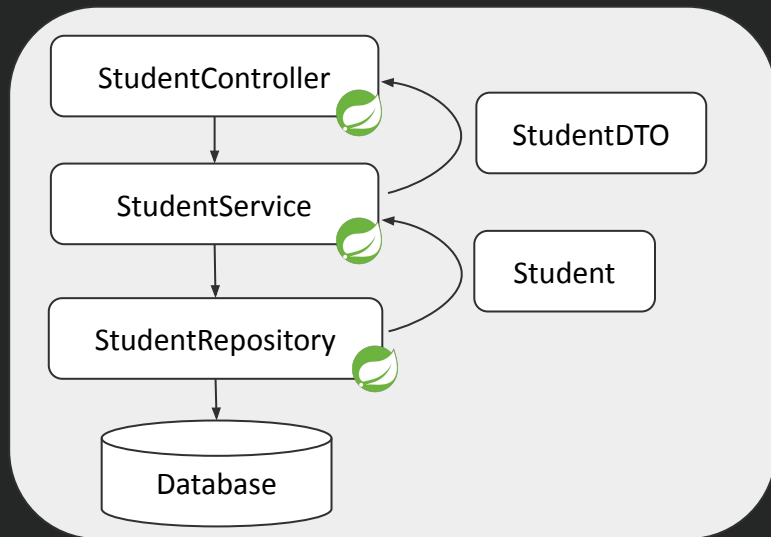
    @Test
    public void shouldMultiplySuccessfully() {
        assertThat(multiply(2,3)).isEqualTo(6);
    }
}
```


Make sure you use the right imports!

MultiplierTest.java

Should you use Dependency Injection everywhere?

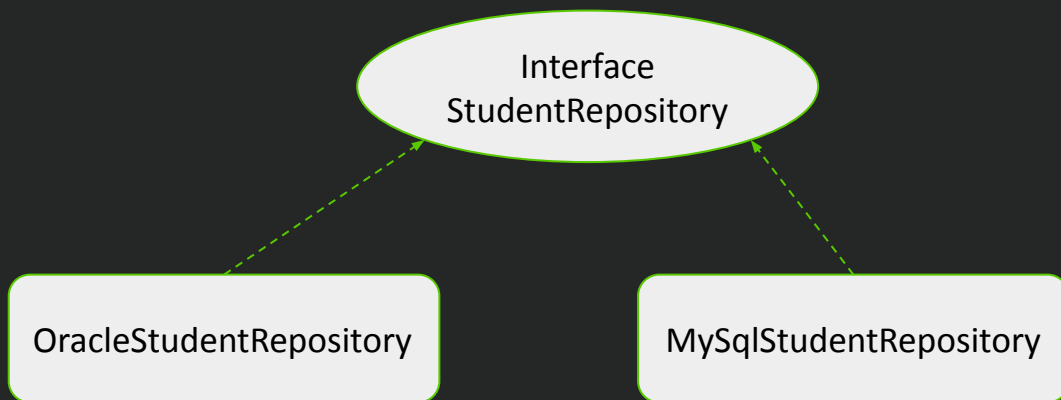
- Typically used for Controller /Service /Repository classes
- Not used for data classes



 : Spring component

Qualifier: which instance to inject

- By default, Dependency Injection is based on type
- How to do when multiple candidates have same type?



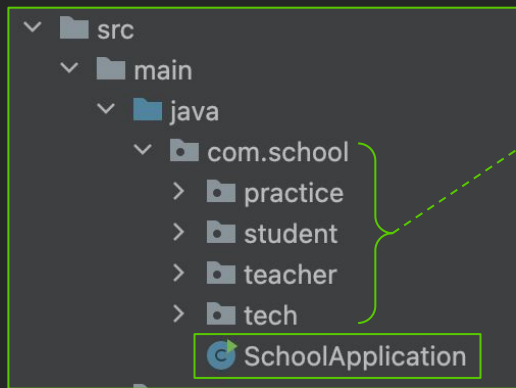
Spring Boot is opinionated

- Default libraries that come with Spring Boot starters:

<i>feature</i>	<i>libraries</i>	<i>Corresponding starters</i>
logging	slf4j, LogBack	all spring-boot-starters
testing	JUnit, assertJ, Mockito	spring-boot-starter-test
Database access	Hibernate (JPA implementation), HikariCP (connection pool)	spring-boot-starter-data-jpa
Web dependencies	Tomcat, Jackson	spring-boot-starter-web

@SpringBootApplication

- Identifies the “main” class for any Spring Boot application
- identifies the base package components to be scanned



```
package com.school;
```

```
@SpringBootApplication
```

```
public class SchoolApplication {
```

```
    public static void main(String[] args) {  
        SpringApplication.run(SchoolApplication.class, args);  
    }  
}
```