

Spring Boot support for Unit Testing



What do you need for Spring Boot unit testing?

- Access to the Spring Application Context
- Support for Spring dependency injection
- Retrieve data from Spring application.properties
- Mock object support for web, data, REST APIs etc ...

Unit Testing support in Spring Boot

- Spring Boot provides rich testing support
- **@SpringBootTest**
 - Loads the application context
 - Support for Spring dependency injection
 - You can access data from Spring application.properties
 - ...

Spring Boot Starter - for Testing support

- Add Maven dependency

pom.xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

Starter includes a transitive
dependency on JUnit 5

We get it for free :-)

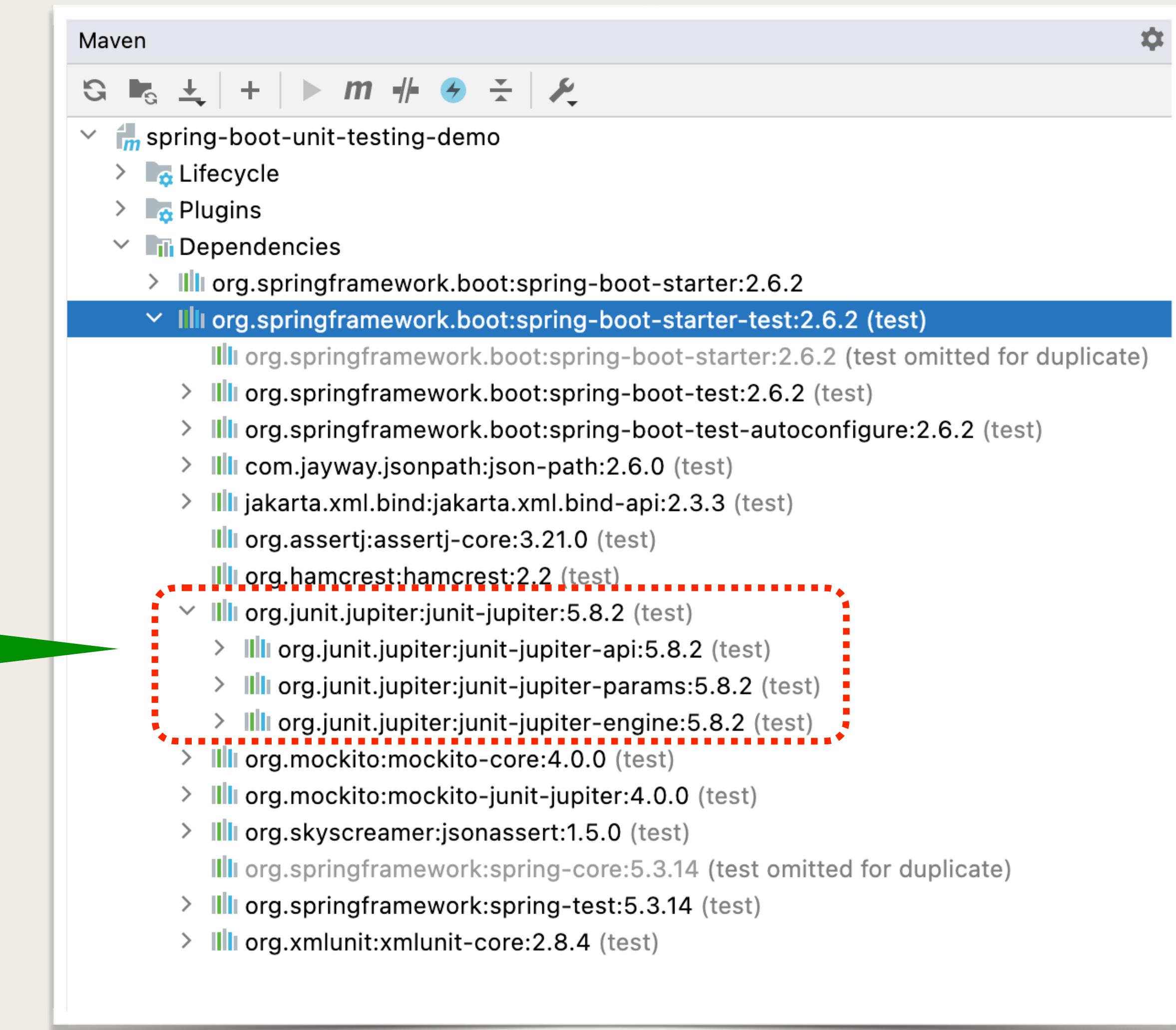
Spring Boot Starter - Transitive Dependency for JUnit 5

pom.xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

Starter includes a transitive dependency on JUnit 5

We get it for free :-)



Spring Boot Starter - Transitive Dependency for JUnit 5

At command-line, type:

Starter includes a transitive dependency on JUnit 5

We get it for free :-)

```
mvn dependency:tree
```

```
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.luv2code:spring-boot-unit-testing-demo >-----
[INFO] Building spring-boot-unit-testing-demo 1.0.0
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-dependency-plugin:3.2.0:tree (default-cli) @ spring-boot-unit-testing-demo ---
[INFO] com.luv2code:spring-boot-unit-testing-demo:jar:1.0.0
...
[INFO] \- org.springframework.boot:spring-boot-starter-test:jar:2.6.2:test
[INFO]     \- org.junit.jupiter:junit-jupiter:jar:5.8.2:test
[INFO]         +- org.junit.jupiter:junit-jupiter-api:jar:5.8.2:test
[INFO]         +- org.junit.jupiter:junit-jupiter-params:jar:5.8.2:test
[INFO]         \- org.junit.jupiter:junit-jupiter-engine:jar:5.8.2:test
...

```

Spring Boot Test

ApplicationExampleTest.java

```
import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
  
@SpringBootTest  
public class ApplicationExampleTest {  
  
    @Test  
    void basicTest() {  
        // ...  
    }  
}
```

Loads Spring Application Context

Inject Spring Beans

ApplicationExampleTest.java

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.beans.factory.annotation.Autowired;

@SpringBootTest
public class ApplicationExampleTest {

    @Autowired
    StudentGrades studentGrades;

    @Test
    void basicTest() {
        // ...
    }
}
```

```
@Component
public class StudentGrades {
    ...
}
```

Injection

Access Application Properties

ApplicationExampleTest.java

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.beans.factory.annotation.Value;

@SpringBootTest
public class ApplicationExampleTest {
    @Value("${info.school.name}")
    private String schoolInfo;
    @Value("${info.app.name}")
    private String appInfo;

    @Test
    void basicTest() {
        // ...
    }
}
```

```
info.school.name=luv2code
info.app.name=My Super Cool Gradebook
```

Access data from
application.properties

Access Application Context

ApplicationExampleTest.java

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;

@SpringBootTest
public class ApplicationExampleTest {

    @Autowired
    ApplicationContext context;

    @Test
    void basicTest() {
        // ...
    }
}
```

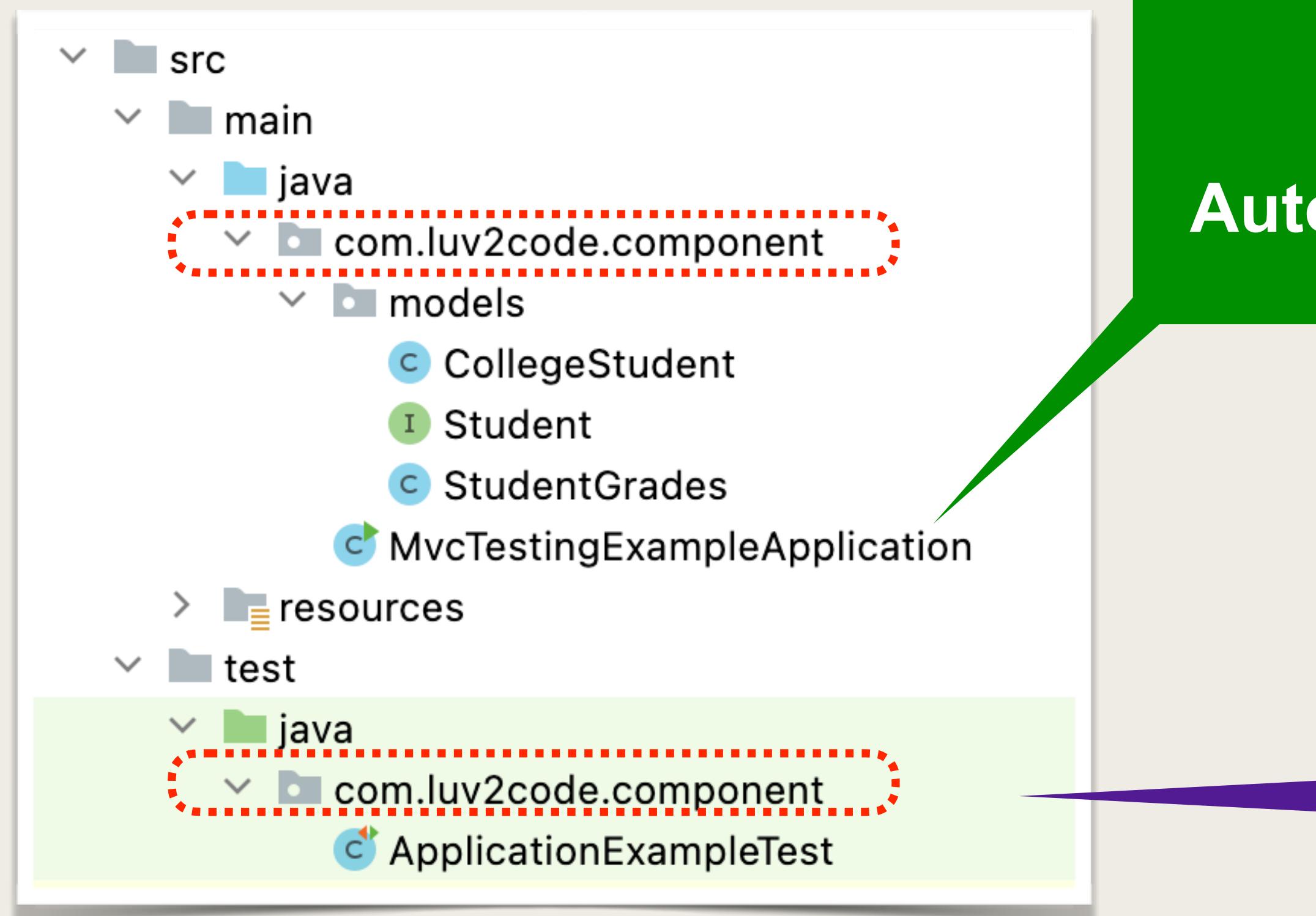
Access Spring Application Context

@SpringBootTest configuration

**Place your test class in test package
same as your main package**

- This implicitly defines a base search
 - Allows you to leverage default configuration
 - No need to explicitly reference the main Spring Boot application class

More on Configuration



Main Spring Boot application class

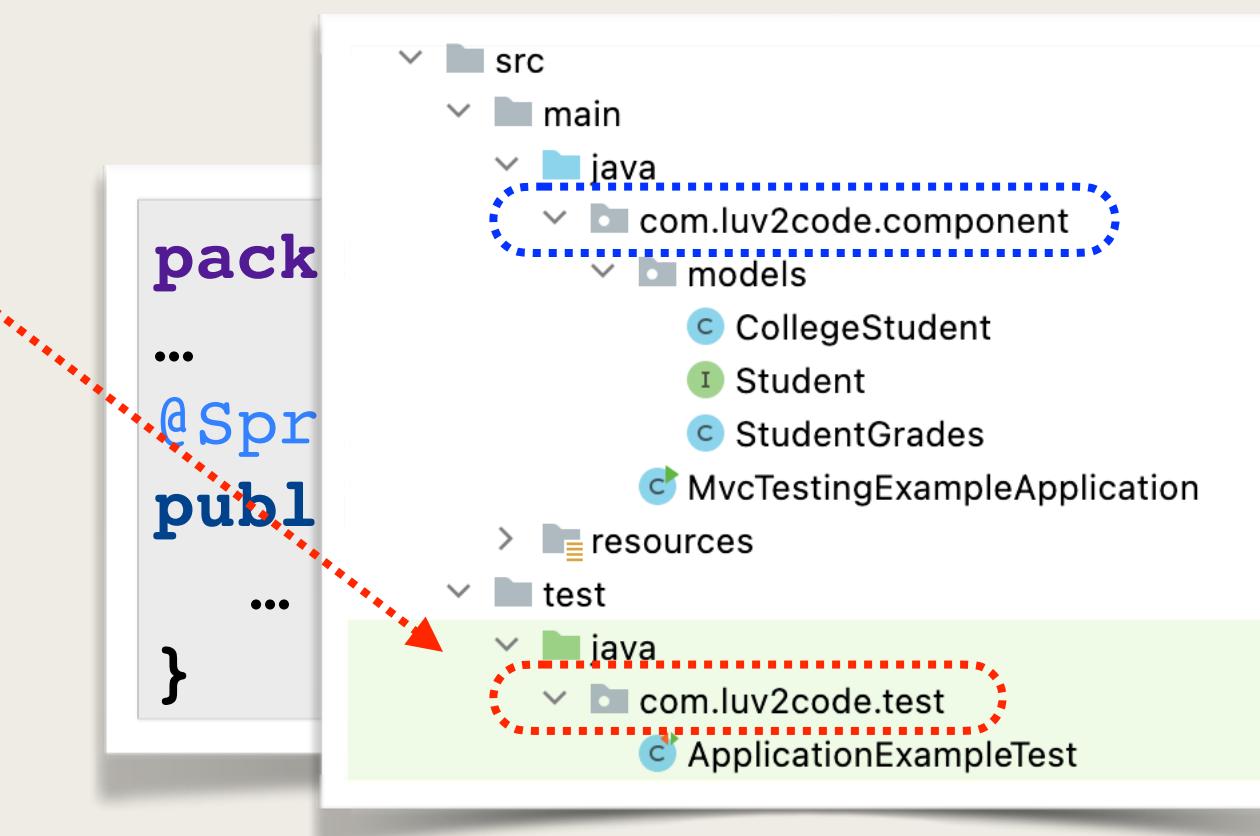
Automatically component scans sub-packages

Same package names

... no additional config required ...

More on Configuration

- Default configuration is fine if everything is under
 - **com.luv2code.component**
- But what if test class is in a different package???
 - **com.luv2code.test**



Explicitly reference
main SpringBoot class

```
MvcTestingExampleApplication.class)
    sampleTest {
```

Our Sample Project

- Student grading system ... code was created by a previous employee
- The code doesn't have any tests ... gasp!
- We've been tasked with developing the unit tests ... yaayy!!!!

