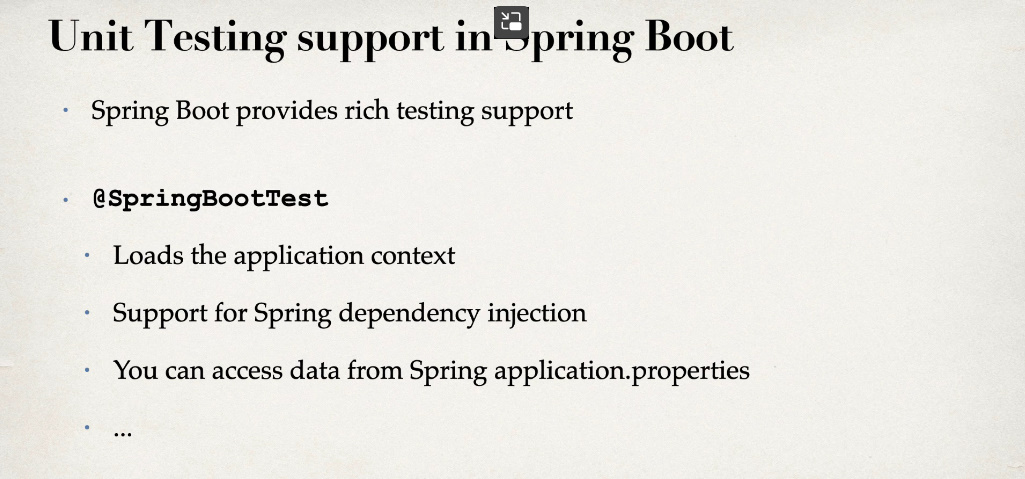
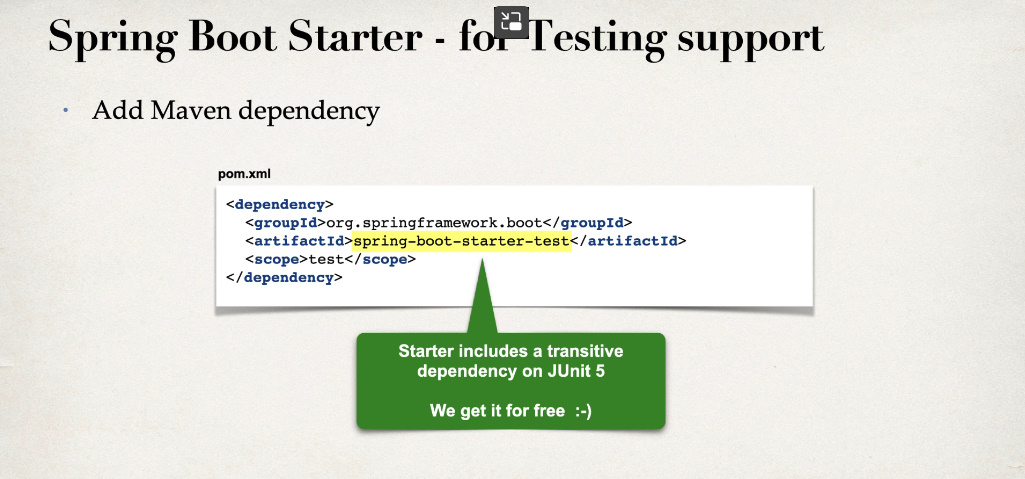
**@SpringBootTest**

Pentru a putea efectua teste asupra codului din Spring Boot, avem nevoie de:

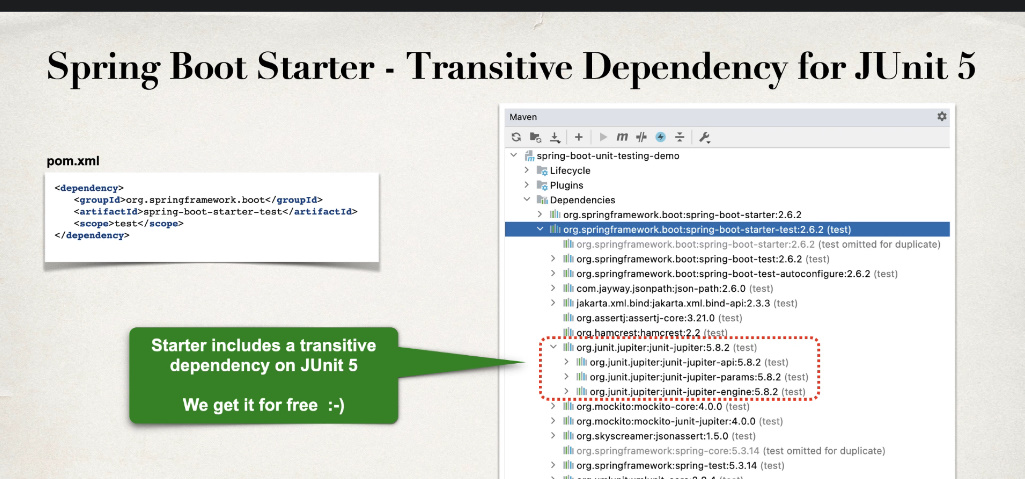
* Acces la Spring Application Context
* Support pentru Spring dependency injection
* Acces la Spring application.properties
* Mock object support pentru web, data, REST APIs etc.
* Acces la toate acestea ni-l ofera anotatia @**SpringBootTest**

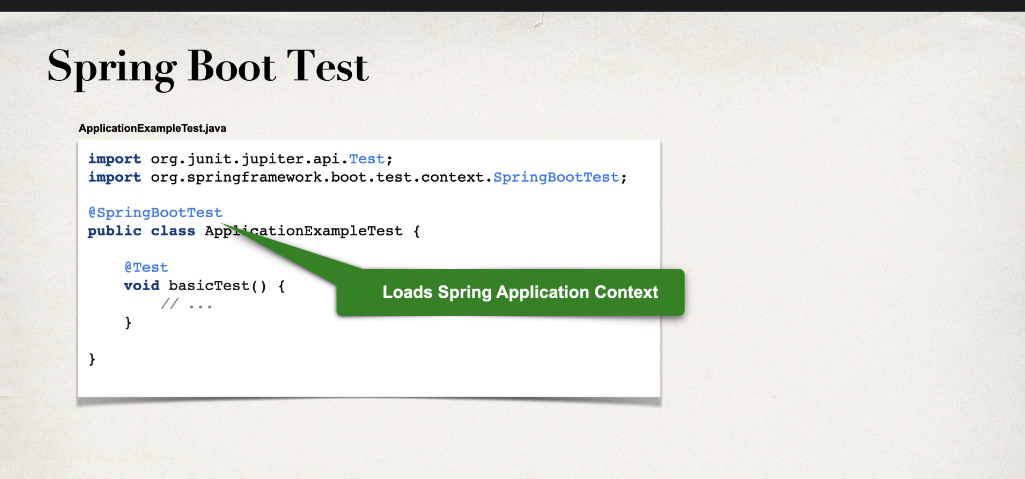


**Dependency**

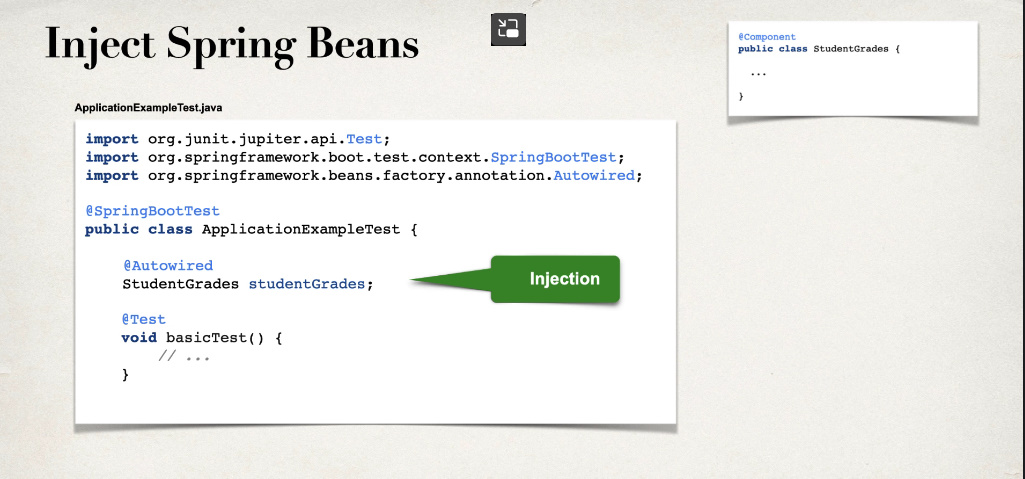


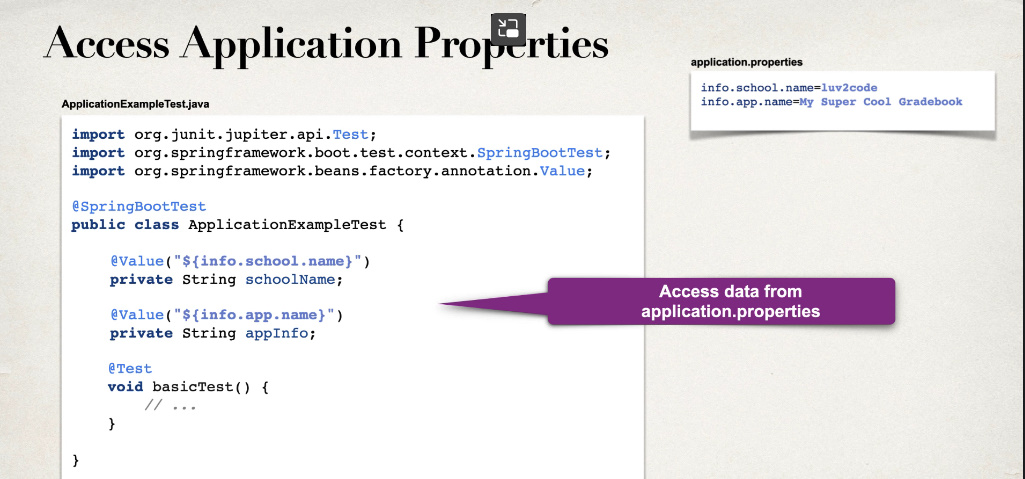
Aceasta dependenta include dependentele necesare pentru Junit



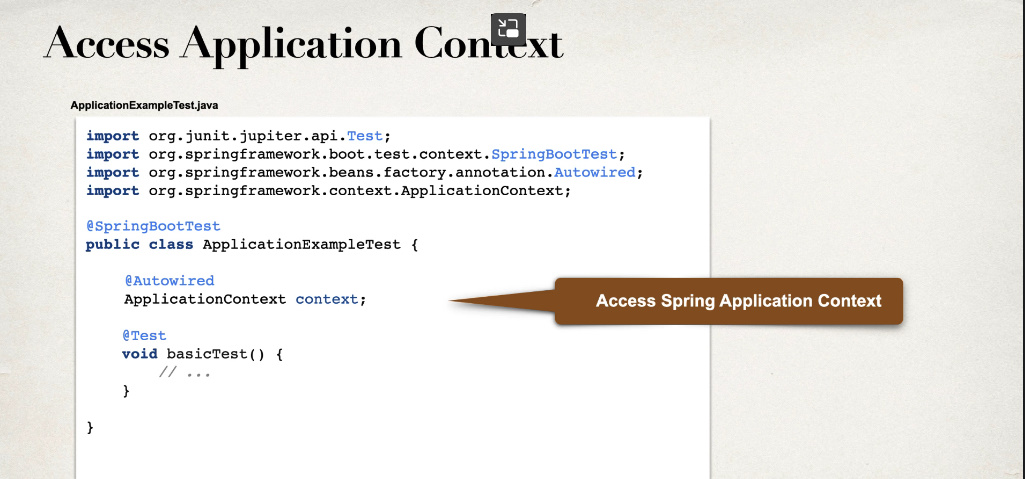


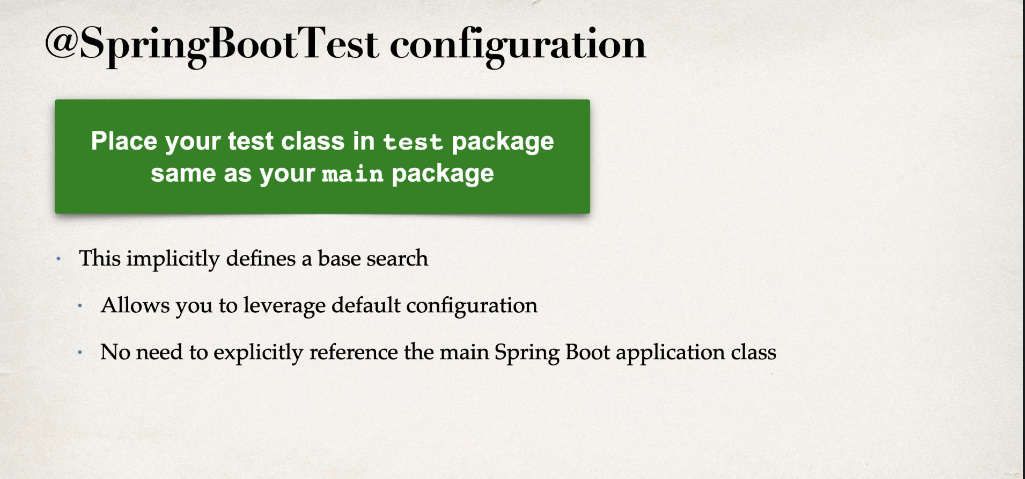
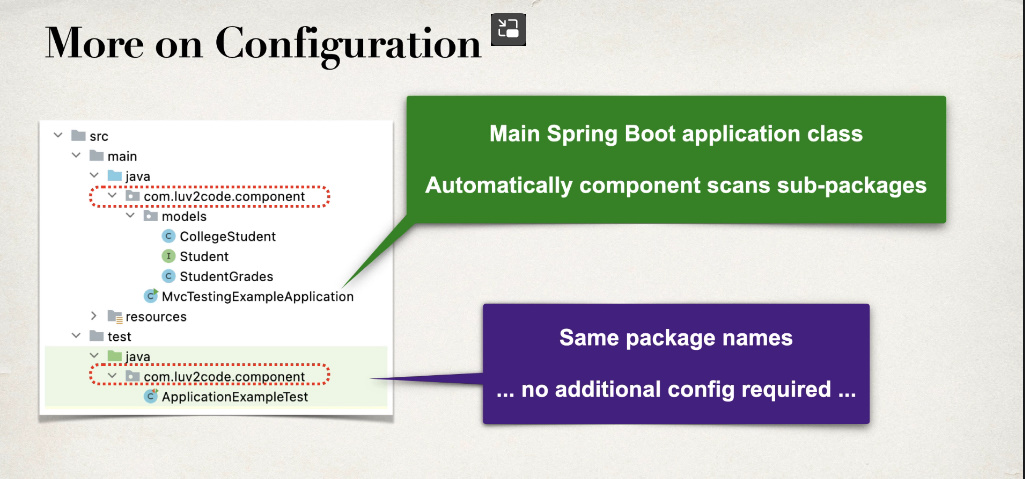
* Acum, anotatia @SpringBootTest ne va oferi posibilitatea de accesa toate beanurile din IoC container create pe baza lui application context, de ex:

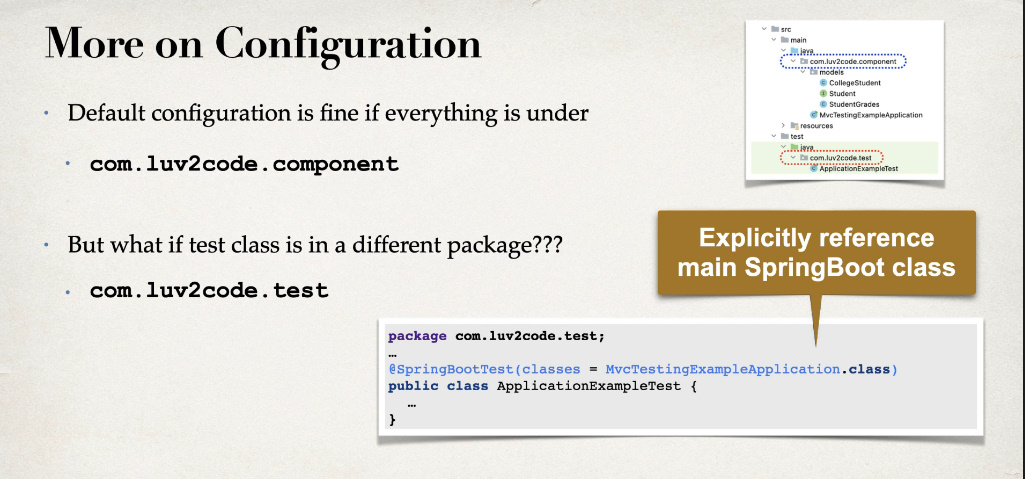


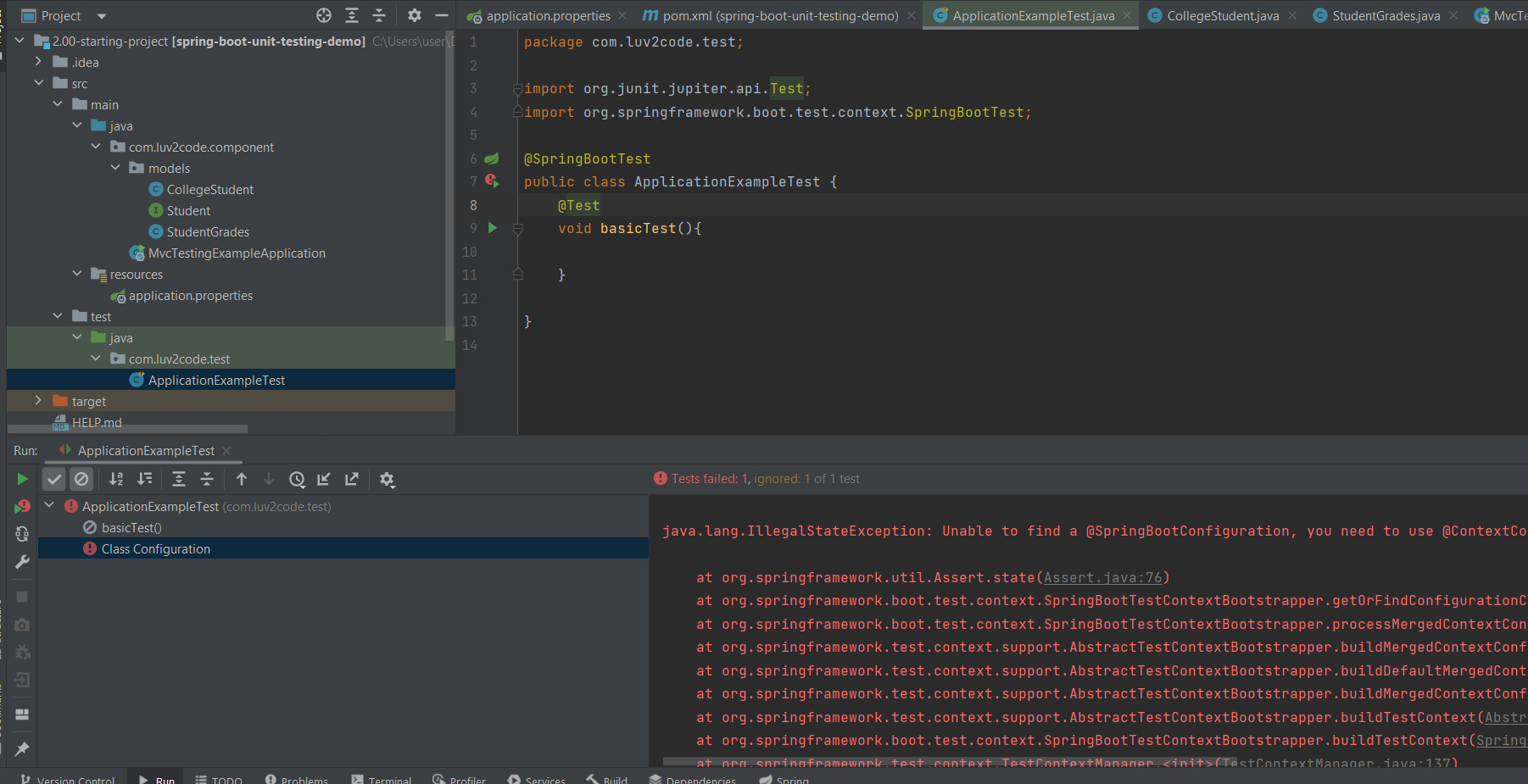


* Putem injecta si ApplicationContext:



* **Pentru ca @SpringBootTest annotation sa functioneze corect, e necesar ca in folderul test sa cream package identic cu cel din java in care se afla clasa ce e application context, dar numele clasei nu conteaza**
* 
* 
* Dar, s-ar putea sa nu vrem sa avem aceleasi package ca si in java, unde se afla clasa app context.In asa caz, va trebui explicit sa scriem clasa care e app config:

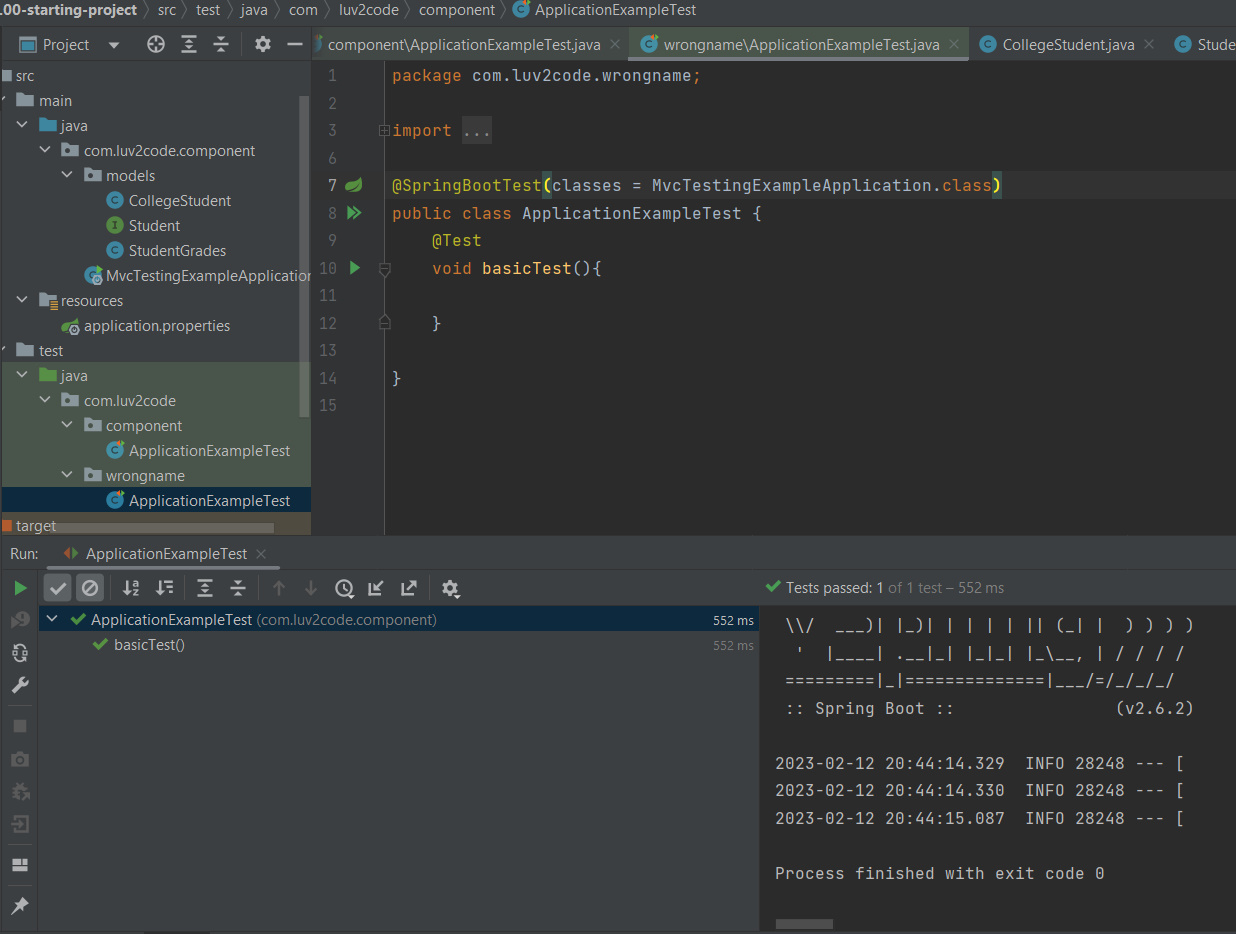


**@SpringBootTest(classes=ClasaAppConfig.class**



**Deci, daca nu punem acelasi nume la package la clasa din test ce foloseste @SpringBootTest ca la cea din app context, vom primi eroare, deoarece nu s-a gasit o clasa in java ce sa aiba acelasi package si sa fie app context**

* Aceasta anotatie va face ca testele sa ruleze ca si acum ar fi deschisa aplicatia





**assertAll(“mesaj de eroare”,()->assert....,()->assert...,...)**

* assertAll() ne ofera posibilitatea de a testa mai multe asserturi deodata, intr-o singura metoda. Fiecare assert va fi o lamba expression. Daca macar un assert da fail, testul da fail