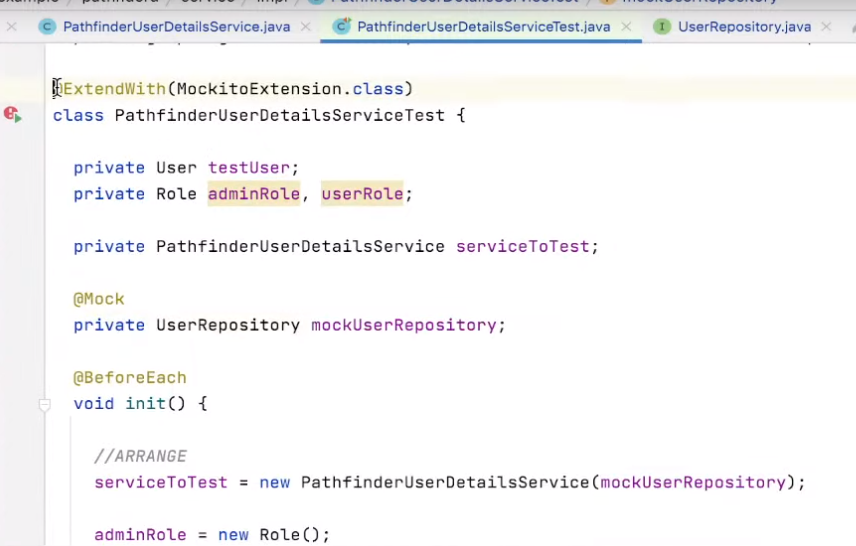
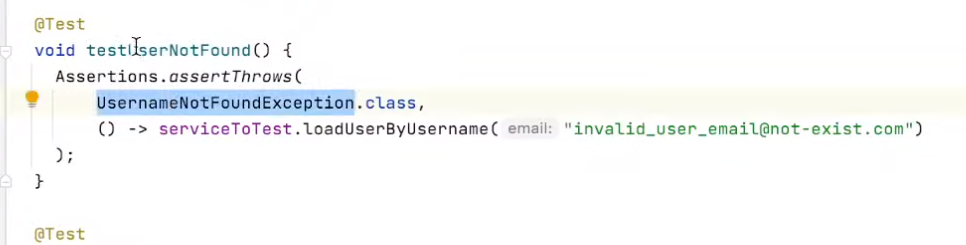
1. **Unit tests with Mockito**

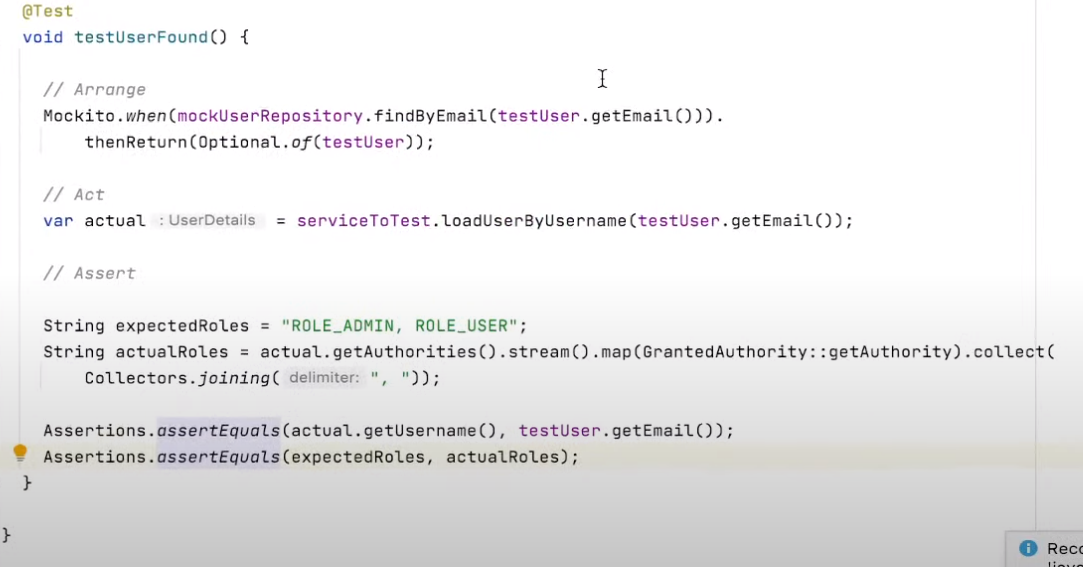
* @ExtendWith(MockitoExtension.class) for mokito functionalities



* Exception test



* Test normal state of class Mockito.when to teach the mocked object (repository)



1. **Test Controller**

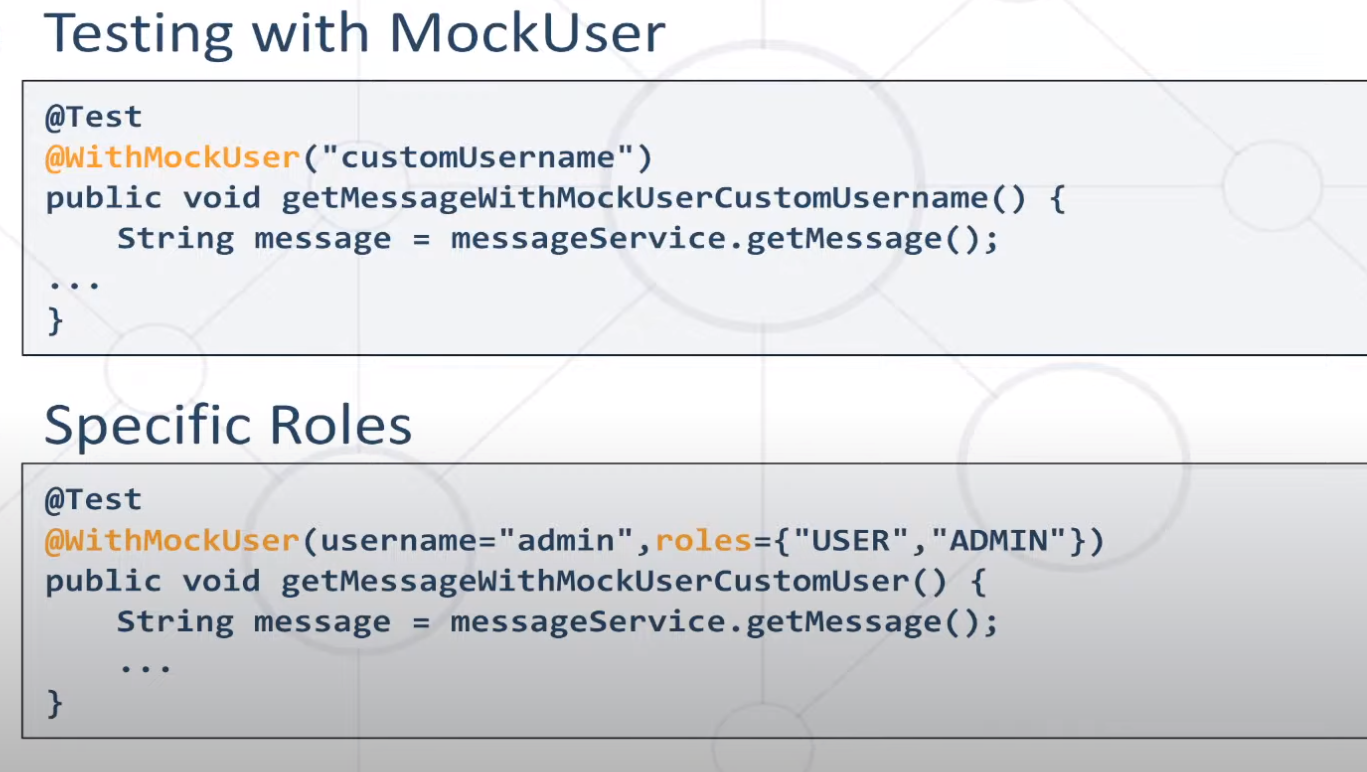
* С @SpringBootTest се правят интеграционните тестове и вдига апликацията
* С MockMvc се пращат заявки към application-a който е вдигнат





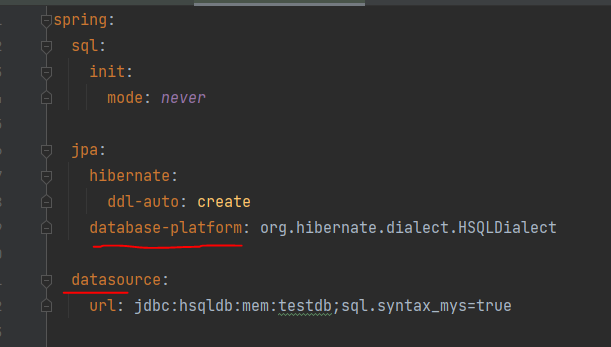
* Когато има Spring Security се използва @WithMockUser за да тества с логнат User
* Dependency:

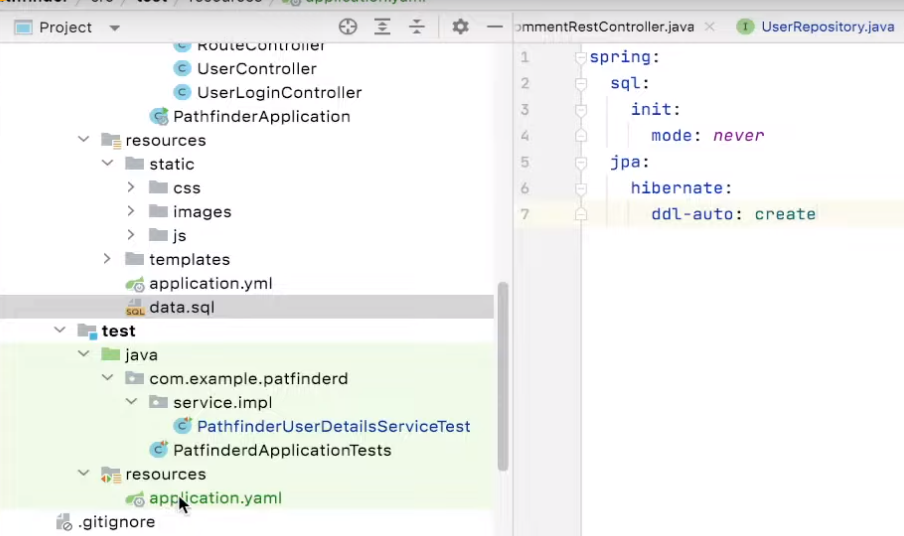




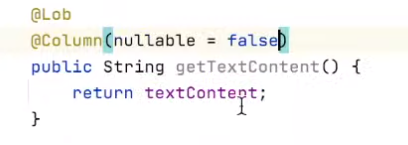
* **Import hsqldb –** in memory DB
* Добавя се в тестовия .yml диалект за HSQLdb



* **New application.yml file in test/java/resources**
* Добавя се spring.jpa.database-platform=org.hibernate.dialect.HSQLDialect
* И datasource.url = jdbc…. За да може да разчита “TEXT” и други типове колони в таблици
* 



* **@Lob annotation** (Large object) **вместо „TEXT”**
* Използва се вместо columndefinition при Entity-тата за да работят интеграционните тестове

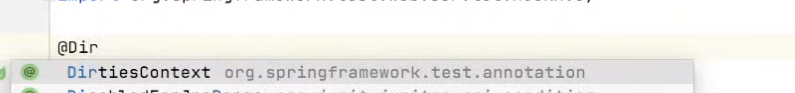


* **Static import**



* **@DirtiesContext**

**-**избягва се да се ползва. Създава отделен контекст за анотирания тестови клас



1. **Pathfinder tests**

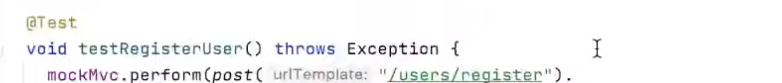
* REST Controller





* Normal Controller





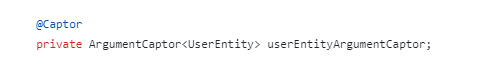


* **Test whether the user is saved**
* Verify(mockedObject).save(any) – checks if “save” method is invoked on userRepository with any arguments.



* **Test that returns the saved User**

Чрез ArgumentCaptor<UserEntity> може да се запише какво е върнал метода .save(userEntityArgumentCaptor.captrue()) и да се достъпи по-нататък в теста с .getValue();

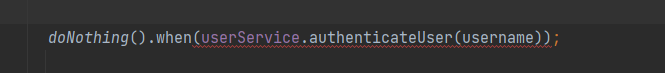




* Втория начин проверява дали подадения като аргумент User съответства на testUser.



* При методи с return type. Няма да се изпълни.



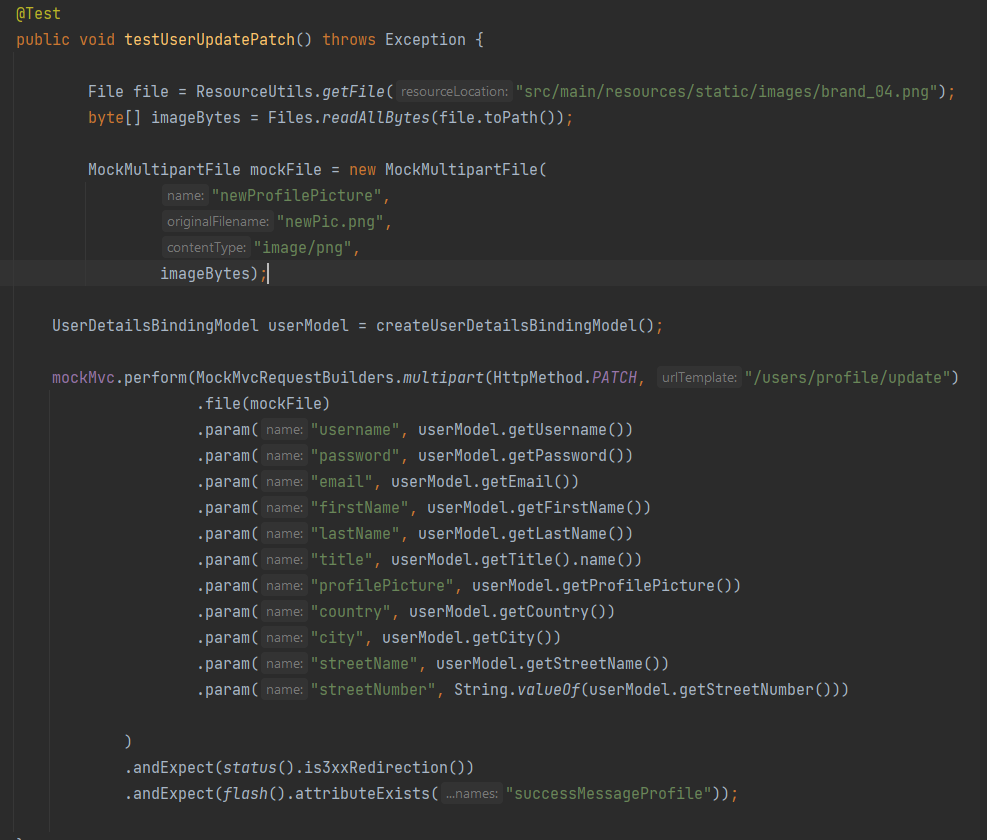
* Тест на model attribute value



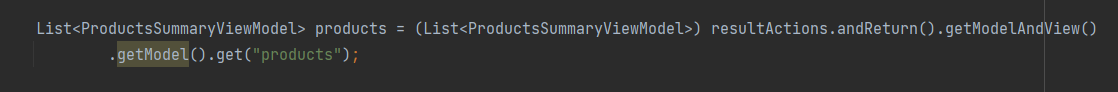
* Test multipart file – multipart(“method”, “URI”)

MockMultipartFile -> file(mockedImage)

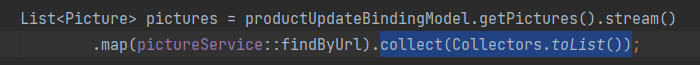
* Empty MockMultipartFIle (new byte[0])
* Attribute – “name” is representing the field name in the binding model (request param)



* Retrieve elements from a list attribute



* Modifiable and unmodifiable list
* This returns modifiable colletion



* toList() returns unmodifiable