I tried to find a way to do a Junit test for my backend code

I found some example code online, as shown below

```
@Test
public void testAddUser() throws Exception {
    // Create a user object for the request body
    User newUser = new User("newuser", "newpassword");

    // Mock behavior of userRepository.save() to return the saved user
    when(userRepository.save(any(User.class))).thenReturn(newUser);

    // Perform a POST request to add a new user
    mockMvc.perform(post("/api/users")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(newUser))) // Convert user object to JSON
string
            .andExpect(status().isCreated()) // Expect HTTP status code 201 Created
            .andExpect(jsonPath("$.username").value("newuser")) // Verify username
            .andExpect(jsonPath("$.password").value("newpassword")); // Verify password
}
```

Another examples that is dealing with get request

```
// Asserting on the Entire Array
mockMvc.perform(get("/api/users")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$").isArray()) // Check if it's an array
        .andExpect(jsonPath("$", hasSize(3))); // Check if it has 3 elements

// Asserting on Individual Objects within the Array
mockMvc.perform(get("/api/users")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$[0].id").value(1)) // Check id of first object
        .andExpect(jsonPath("$[0].name").value("Alice")); // Check name of first object
```

This do this I need to import a lot of classes

Also add dependency to my code, to the xml file

```
<dependencies>
    <!-- Other dependencies -->

    <!-- Spring Boot Test dependency -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Overall, I need to mock the behavior of the repository class, and use mockMvc.perform to test the API in the controller class.

I did even more research today, and for some reason the above method always return an empty response body, so I decided to change to another method

```
Item item = new Item(11L, "name", 1.1);
when(itemsRepository.findByItemId(11L)).thenReturn(item);
//get the getItem method in the controller
ResponseEntity<?> response = this.itemsController.getItem(11L, "name");
assertEquals(HttpStatus.OK, response.getStatusCode());
assertNotNull(response.getBody());
assertEquals(true, response.getBody().equals(item));
```

I decided to directly call the method in the controller, and check the return of the method.