1. اختبار الخادم باستخدام مكتبة Junit 5

JUnit هي واحدة من أكثر أطر اختبار الوحدات شيوعًا في نظام Java البيئي. يحتوي إصدار JUnit 5 على عدد من الابتكارات المثيرة، بهدف دعم الميزات الجديدة في Java 8 وما فوق، بالإضافة إلى تمكين العديد من أنماط الاختبار المختلفة.

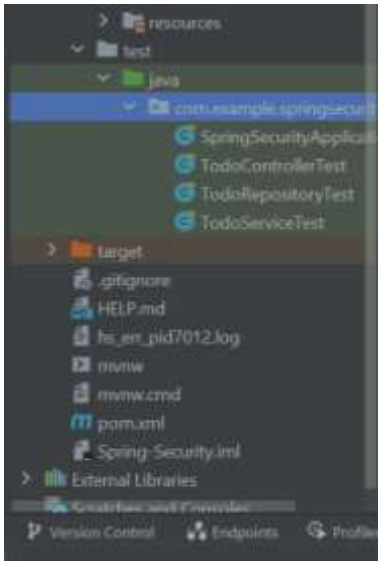
يعد إعداد JUnit 5.x.0 أمرًا بسيطًا جدًا، نحتاج فقط إلى إضافة التبعية التالية إلى ملف pom.xml الخاص بنا:

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

<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <scope>test</scope>
</dependency>
```

ويمكننا اختبار كل وحدة في داخل كل طبقة في المشروع، في البداية سنقوم بإنشاء test class لكل class نود اختباره في داخل مجلد test الموجود بشكل تلقائي داخل المشروع



1.1 مثال لاختبار طبقة التحكم (Controller layer)

سنقوم باختبار هذه الوحدة:

```
@GetMapping("/get-all")
public ResponseEntity<List<Todo>> getAllTodos(){
    return
ResponseEntity.status(HttpStatus.OK).body(todoService.getTodos());
}
```

في البداية سنقوم بإنشاء كلاس TodoControllerTest داخل مجلد test ثم سنقوم بكتابة التعليقات التوضيحية الخاصة بهذا الكلاس ليتم اختباره بشكل سليم مع وضع قيم يتم اختبارها وكأنها موجودة بداخل قاعدة البيانات لتساعد في اختبار الوحدات باستخدام التعليق التوضيحي @BeforeEach وكذلك استخدام تعليق @MockBean للتعامل مع طبقة الخادم المتصلة بطبقة التحكم دون الخوض في تفاصيلها

```
@ExtendWith(SpringExtension.class)
@WebMvcTest(value = TodoController.class , excludeAutoConfiguration =
{SecurityAutoConfiguration.class})
public class TodoControllerTest {
    @MockBean
    TodoService todoService;

    @Autowired
    MockMvc mockMvc;

    Todo todo1,todo2,todo3;
    MyUser myUser;

    ApiResponse apiResponse;

    List<Todo> todos,todoList;

    @BeforeEach
    void setUp() {
        myUser=new MyUser(1,"Maha" , "12345" , "ADMIN" , null);
        todo1 = new Todo(1, "todo1", myUser );
        todo2 = new Todo(2 , "todo2",  myUser );
        todo3 = new Todo(3 , "todo3", myUser );
        todos= Arrays.asList(todo1);
        todoList=Arrays.asList(todo2);



    }
```

والآن سنقوم بإنشاء دالة تقوم باختبار الوحدة السابقة مع كتابة التعليق التوضيحي @Test لتفعيل خاصية الاختبار

```
@Test
public void GetAllTodo() throws Exception {
  Mockito.when(todoService.getTodos()).thenReturn(todos);
  mockMvc.perform(get("/api/v1/todo/get-all"))
   .andExpect(status().isOk())
   .andExpect(MockMvcResultMatchers.jsonPath("$",
Matchers.hasSize(1)))
.andExpect(MockMvcResultMatchers.jsonPath("$[0].title").value("todo1"
));
}
```

ثم سنقوم بتشغيل زر التشغيل Run الذي بجانب الدالة لنتأكد أن الاختبار يعمل وأن حالته success

1.2 مثال لاختبار طبقة الخدمة (Service layer)

سنقوم باختبار هذه الوحدة:

```
@PostMapping()
public ResponseEntity <ApiResponse> addTodos(@AuthenticationPrincipal
MyUser myUser, @RequestBody Todo todo){
    todoService.addTodo(myUser.getId(),todo);
    return ResponseEntity.status(HttpStatus.OK).body(new
ApiResponse("New Todo added !",201));
}
```

في البداية سنقوم بإنشاء كلاس TodoServiceTest داخل مجلد test ثم سنقوم بكتابة التعليقات التوضيحية الخاصة بهذا الكلاس ليتم اختباره بشكل سليم مع وضع قيم يتم اختبارها وكأنها موجودة بداخل قاعدة البيانات لتساعد في اختبار الوحدات باستخدام التعليق التوضيحي @BeforeEach وكذلك استخدام تعليق @Mock للتعامل مع طبقة المستودع المتصلة بطبقة الخدمة دون الخوض في تفاصيلها

```
@ExtendWith(MockitoExtension.class)
public class TodoServiceTest {

    @InjectMocks
    TodoService todoService;
    @Mock
    TodoRepository todoRepository;
    @Mock
    AuthRepository authRepository;

    MyUser user;

    Todo todo1,todo2,todo3;

    List<Todo> todos;

    @BeforeEach
    void setUp() {
        user=new MyUser(null,"majd","123","Admin", null);
        todo1=new Todo(null,"todo1",user);
        todo2=new Todo(null,"todo2",user);
        todo3=new Todo(null,"todo3",null);

        todos=new ArrayList<>();
        todos.add(todo1);
        todos.add(todo2);
        todos.add(todo3);
    }
```

والآن سنقوم بإنشاء دالة تقوم باختبار الوحدة السابقة مع كتابة التعليق التوضيحي @Test لتفعيل خاصية الاختبار

```
@Test
public void AddTodoTest(){


when(authRepository.findMyUserById(user.getId())).thenReturn(user);

    todoService.addTodo(user.getId(),todo3);
    verify(authRepository,times(1)).findMyUserById(user.getId());
    verify(todoRepository,times(1)).save(todo3);
}
```

1.3 مثال لاختبار وحدة في طبقة المستودع (repository layer)

سنقوم باختبار هذه الوحدتين

```
public interface TodoRepository extends JpaRepository<Todo,Integer> {
    Todo findTodoById(Integer id);

    List<Todo> findAllByMyUser(MyUser myUser);
}
```

في البداية سنقوم بإنشاء كلاس TodoRepositoryTest داخل مجلد test ثم سنقوم بكتابة التعليقات التوضيحية الخاصة بهذا الكلاس ليتم اختباره بشكل سليم مع وضع قيم يتم اختبارها وكأنها موجودة بداخل قاعدة البيانات لتساعد في اختبار الوحدات باستخدام التعليق التوضيحي @BeforeEach

```
@ExtendWith(SpringExtension.class)
@DataJpaTest
@AutoConfigureTestDatabase(replace
=AutoConfigureTestDatabase.Replace.NONE )
public class TodoRepositoryTest {

    @Autowired
    TodoRepository todoRepository;

    MyUser user;

    Todo todo1,todo2,todo3;

    List<Todo> todos;

    @BeforeEach
    void setUp() {
       user=new MyUser(null,"majd","123","Admin", null);
       todo1=new Todo(null,"todo1",user);
       todo2=new Todo(null,"todo2",user);
       todo3=new Todo(null,"todo3",user);

    }
```

والآن سنقوم بإنشاء دالتين تقوم باختبار الوحدات السابقة مع كتابة التعليق التوضيحي Test@ لتفعيل خاصية الاختبار

```
@Test
public void findTodoByIdTest(){

    todoRepository.save(todo1);
    Todo todo=todoRepository.findTodoById(todo1.getId());
    Assertions.assertThat(todo).isEqualTo(todo1);

}
@Test
public void findAllByMyUserTest(){

    todoRepository.save(todo1);
    todoRepository.save(todo2);
    todoRepository.save(todo3);

    List<Todo> todoList=todoRepository.findAllByMyUser(user);


Assertions.assertThat(todoList.get(0).getMyUser().getId()).isEqualTo(
user.getId());

}
```