TP N°3 : Les tests unitaires et les tests d’intégration avec Spring BOOT, JUNIT et MOCKITO

SOMMAIRE

I- Objectifs :

- ✓ Apprendre comment réaliser les tests unitaires.
- ✓ Apprendre comment réaliser les tests d'intégration.

II- Outils utilisés :

- ✓ JDK 1.8 ;
- ✓ Eclipse avec le plugin Maven ;
- ✓ Connection Internet pour télécharger les dépendances (Spring Boot, …).

III- Pré requis :

Nous allons réaliser les tests unitaires et les tests d'intégration de l'application développée dans le TP n° 10.

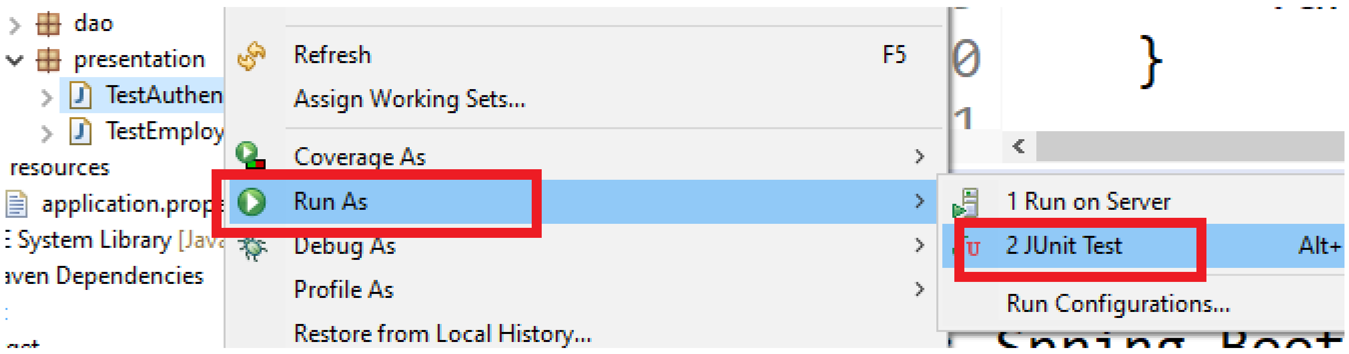
III- Développement des tests unitaires

a. Tester le contrôleur AuthenticationController

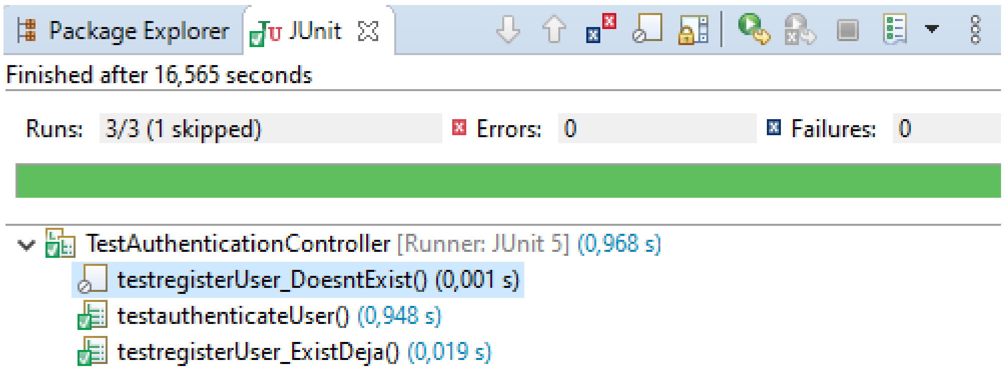
- Créer la classe suivante :

```java
package ma.formations.unitaire.presentation;

import static org.mockito.Mockito.doNothing;
import static org.mockito.Mockito.when;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import java.util.Arrays;

import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import org.springframework.test.web.servlet.MockMvc;

import com.fasterxml.jackson.databind.ObjectMapper;

import ma.formations.controller.AuthenticationController;
import ma.formations.domaine.UserVo;
import ma.formations.jwt.AuthEntryPointJwt;
import ma.formations.jwt.JwtUtils;
import ma.formations.service.IEmpService;
import ma.formations.service.IUserService;

@ExtendWith(SpringExtension.class)
@WebMvcTest(AuthenticationController.class)
public class TestAuthenticationController {

        @Autowired
        private MockMvc mockMvc;

        @MockBean
        AuthenticationManager authenticationManager;

        @MockBean
        private IUserService userService;

        @MockBean
        private JwtUtils jwtUtils;

        @MockBean
        AuthEntryPointJwt authEntryPointJwt;

        @MockBean
        IEmpService empService;

        @Test
        void testauthenticateUser() throws Exception {
                String tokenTest = "AAAA.BBBB.SSSS";
                UserVo userVoTest = new UserVo();
                userVoTest.setUsername("admin");
                userVoTest.setPassword("admin");

Authentication authenticationResult = new UsernamePasswordAuthenticationToken(userVoTest.getUsername(),
        userVoTest.getPassword(), Arrays.asList(new SimpleGrantedAuthority("ADMIN")));

when(authenticationManager.authenticate(Mockito.any())).thenReturn(authenticationResult);
when(jwtUtils.generateJwtToken(Mockito.any())).thenReturn(tokenTest);
mockMvc.perform(post("/auth/signin").content(asJsonString(userVoTest)).contentType(MediaType.APPLICATION
_JSON).accept(MediaType.APPLICATION_JSON)).andExpect(status().isOk())
       .andExpect(jsonPath("$.jwttoken").value(tokenTest))
       .andExpect(jsonPath("$.username").value(userVoTest.getUsername()))
       .andExpect(jsonPath("$.roles[0]").value("ADMIN"));
}

        @Test
        void testregisterUser_ExistDeja() throws Exception {
                UserVo userVoTest = new UserVo();
                userVoTest.setUsername("admin");
                userVoTest.setPassword("admin");
                when(userService.existsByUsername(userVoTest.getUsername())).thenReturn(true);

        mockMvc.perform(post("/auth/signup").content(asJsonString(userVoTest)).contentType(MediaType.APPL
ICATION_JSON)
                                        .accept(MediaType.APPLICATION_JSON))
                    .andExpect(status().isBadRequest())
                    .andExpect(jsonPath("$").value("Error: Username is already taken!"));
          }

          @Test
          @Disabled
          void testregisterUser_DoesntExist() throws Exception {
                    UserVo userVoTest = new UserVo();
                    userVoTest.setUsername("admin");
                    userVoTest.setPassword("admin");

                    when(userService.existsByUsername(userVoTest.getUsername())).thenReturn(false);
                    doNothing().when(userService).save(userVoTest);


          mockMvc.perform(post("/auth/signup").content(asJsonString(userVoTest)).contentType(MediaType.APPL
ICATION_JSON)
                                        .accept(MediaType.APPLICATION_JSON)).
                    andExpect(status().isOk())
                    .andExpect(jsonPath("$").value("User registered successfully!"));
          }

          public static String asJsonString(final Object obj) {
                    try {
                              return new ObjectMapper().writeValueAsString(obj);
                    } catch (Exception e) {
                              throw new RuntimeException(e);
                    }
          }
}
```

- Pour tester la classe, cliquer à droite de la souris sur la classe, ensuite ***Run As ->JUNIT Test*** comme le montre l'écran suivant :



Le résultat devrait être :



```
Finished after 16,565 seconds
Runs: 3/3 (1 skipped)    Errors: 0    Failures: 0

TestAuthenticationController [Runner: JUnit 5] (0,968 s)
    testregisterUser_DoesntExist() (0,001 s)
    testauthenticateUser() (0,948 s)
    testregisterUser_ExistDeja() (0,019 s)
```

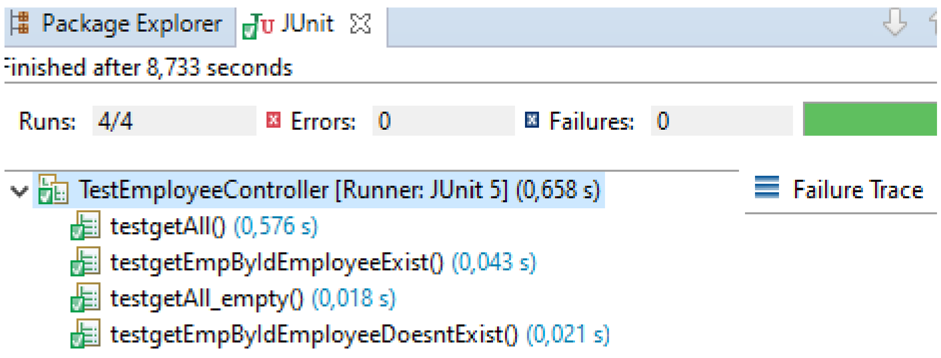
b. Tester le contrôleur EmpController

- Créer la classe suivante :

```
package ma.formations.unitaire.presentation;

import static org.mockito.Mockito.when;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import java.util.Arrays;
import java.util.List;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.FilterType;
import org.springframework.http.MediaType;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.test.context.support.WithMockUser;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import org.springframework.test.web.servlet.MockMvc;

import ma.formations.controller.EmpController;
import ma.formations.domaine.EmpVo;
import ma.formations.jwt.AuthEntryPointJwt;
import ma.formations.jwt.JwtUtils;
import ma.formations.service.IEmpService;
import ma.formations.service.IUserService;

@ExtendWith(SpringExtension.class)
@WebMvcTest(controllers = EmpController.class,
                                    useDefaultFilters = false,
                                    includeFilters = {@ComponentScan.Filter(type = FilterType.ASSIGNABLE_TYPE, value =
EmpController.class) })
@WithMockUser(authorities = {"ADMIN"},password = "admin1",username = "admin1")
public class TestEmployeeController {
            @MockBean
```

```
        private IEmpService service;

        @Autowired
        private MockMvc mvc;

        @MockBean
        AuthenticationManager authenticationManager;

        @MockBean
        private IUserService userService;

        @MockBean
        private JwtUtils jwtUtils;

        @MockBean
        AuthEntryPointJwt authEntryPointJwt;

        @Test
        void testgetAll() throws Exception {
                List<EmpVo> employees = Arrays.asList(
                                new EmpVo("emp1", 10000d, "Fonction1"),
                                new EmpVo("emp2", 20000d, "Fonction2"),
                                new EmpVo("emp", 30000d, "Fonction3"));
                when(service.getEmployees()).thenReturn(employees);

        mvc.perform(get("/employees").contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JS
ON))
                                .andExpect(status().isOk())
                                .andExpect(jsonPath("$[0].name").value("emp1"))
                                .andExpect(jsonPath("$[1].fonction").value("Fonction2"))
                                .andExpect(jsonPath("$[1].salary").value(20000d))
                                .andExpect(jsonPath("$[2].salary").value(30000d));
        }

        @Test
        void testgetAll_empty() throws Exception {
                when(service.getEmployees()).thenReturn(null);

        mvc.perform(get("/employees").contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JS
ON))
                 .andExpect(status().isOk())
                 .andExpect(jsonPath("$").doesNotExist());
        }
        @Test
        void testgetEmpByIdEmployeeExist() throws Exception {
                EmpVo empTest=new EmpVo();
                empTest.setId(1l);
                empTest.setFonction("INGENIEUR");
                empTest.setSalary(10000d);
                empTest.setName("Foulane");

                when(service.getEmpById(Mockito.any())).thenReturn(empTest);
                mvc.perform(get("/employees/{id}",1L)
                                .contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(jsonPath("$.id").value(empTest.getId()))
                .andExpect(jsonPath("$.name").value(empTest.getName()))
                .andExpect(jsonPath("$.fonction").value(empTest.getFonction()))
                .andExpect(jsonPath("$.salary").value(empTest.getSalary()));
        }

        @Test

        void testgetEmpByIdEmployeeDoesntExist() throws Exception {

                when(service.getEmpById(Mockito.any())).thenReturn(null);
                mvc.perform(get("/employees/{id}",1L)
                                .contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(jsonPath("$").value("employee doen't exist"));
        }
}
```

Les résultats de test de cette classe sont comme suit :


```
Package Explorer   JUnit
Finished after 8,733 seconds
Runs: 4/4    Errors: 0    Failures: 0

TestEmployeeController [Runner: JUnit 5] (0,658 s)        Failure Trace
    testgetAll() (0,576 s)
    testgetEmpByIdEmployeeExist() (0,043 s)
    testgetAll_empty() (0,018 s)
    testgetEmpByIdEmployeeDoesntExist() (0,021 s)
```

c. Tester la couche DA : UserRepository

Créer la classe suivante :

```
package ma.formations.unitaire.dao;

import static org.assertj.core.api.Assertions.assertThat;
import static org.junit.jupiter.api.Assertions.assertNotNull;

import java.util.Arrays;

import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.context.annotation.Import;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.test.context.junit.jupiter.SpringExtension;

import ma.formations.dao.UserRepository;
import ma.formations.service.EmpServiceImpl;
import ma.formations.service.UserServiceImpl;
```

```java
import ma.formations.service.model.Role;
import ma.formations.service.model.User;

@ExtendWith(SpringExtension.class)
@DataJpaTest
@AutoConfigureTestDatabase(replace=Replace.NONE)
@Import(value= {UserServiceImpl.class,BCryptPasswordEncoder.class,EmpServiceImpl.class})
public class TestUserRepository {
	@Autowired
	UserRepository userRepository;
	static User userTest = new User();

	@BeforeAll
	static void init() {
		userTest.setUsername("test123");
		userTest.setPassword("P@sw@rd");
		userTest.setRoles(Arrays.asList(new Role("ADMIN")));
	}

	@BeforeEach
	void save() {
		userRepository.save(userTest);
	}

	@Test
	void testfindByUsername() {
		assertNotNull(userRepository.findByUsername(userTest.getUsername()));

	assertThat((userRepository.findByUsername(userTest.getUsername())).getUsername()).isEqualTo(userTest.getUsern
ame());

	assertThat((userRepository.findByUsername(userTest.getUsername())).getPassword()).isEqualTo(userTest.getPassw
ord());

	}
	@Test
	void testexistsByUsername() {
		assertThat(userRepository.existsByUsername(userTest.getUsername())).isTrue();
	}

	@Test
	void testDoesntexistsByUsername() {
		assertThat(userRepository.existsByUsername("wrong name")).isFalse();
	}
}
```

III- Développement des tests d’intégration

a. Tester la couche présentation avec TestRestTemplate

- Créer la classe suivante pour tester le contrôlleur AuthenticationController :

```
package ma.formations.integration.presentation;

import static org.assertj.core.api.Assertions.assertThat;

import java.util.Arrays;
import java.util.List;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.boot.web.server.LocalServerPort;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import ma.formations.domaine.RoleVo;
import ma.formations.domaine.TokenVo;
import ma.formations.domaine.UserVo;
import ma.formations.service.IUserService;

@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
class TestAuthenticationController {

	@Autowired
	private IUserService userService;

	@LocalServerPort
	private int port;

	@Autowired
	private TestRestTemplate restTemplate;

	@Test
	public void testauthenticateUserIsNotNull() throws Exception {

		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { MediaType.APPLICATION_JSON }));
		// Request to return JSON format
		headers.setContentType(MediaType.APPLICATION_JSON);
		UserVo user = new UserVo();
		user.setUsername("admin1");
		user.setPassword("admin1");
		HttpEntity<UserVo> entity = new HttpEntity<UserVo>(user, headers);
		// TokenVo tokenTest=new TokenVo()
		assertThat(this.restTemplate.exchange("http://localhost:" + port + "/auth/signin", HttpMethod.POST,
entity,
				TokenVo.class)).isNotNull();
	}

	@Test
	public void testauthenticateUserHasToken() throws Exception {
		userService.save(new RoleVo("ADMIN"));
		userService.save(new RoleVo("CLIENT"));
		RoleVo roleAdmin = userService.getRoleByName("ADMIN");
		RoleVo roleClient = userService.getRoleByName("CLIENT");
		UserVo admin1 = new UserVo("admin1", "admin1", Arrays.asList(roleAdmin));
		UserVo admin2 = new UserVo("admin2", "admin2", Arrays.asList(roleAdmin));
		UserVo client1 = new UserVo("client1", "client1", Arrays.asList(roleClient));
		UserVo client2 = new UserVo("client2", "client2", Arrays.asList(roleClient));
		userService.save(admin1);
		userService.save(client1);
		userService.save(client2);
		userService.save(admin2);
		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { MediaType.APPLICATION_JSON }));
		// Request to return JSON format
		headers.setContentType(MediaType.APPLICATION_JSON);
		UserVo user = new UserVo();
		user.setUsername("admin1");
		user.setPassword("admin1");
		HttpEntity<UserVo> entity = new HttpEntity<UserVo>(user, headers);
		// TokenVo tokenTest=new TokenVo()
		ResponseEntity<TokenVo> response = this.restTemplate.exchange("http://localhost:" + port +
"/auth/signin",
				HttpMethod.POST, entity, TokenVo.class);
		assertThat(response.getBody()).isNotNull();
		assertThat(((TokenVo) response.getBody()).getJwttoken()).isNotNull();
		assertThat(((TokenVo) response.getBody()).getRoles()).isNotEmpty();
	}
//
	@Test
	public void testauthenticateUserHasRole() throws Exception {

		userService.save(new RoleVo("ADMIN"));
		userService.save(new RoleVo("CLIENT"));

		RoleVo roleAdmin = userService.getRoleByName("ADMIN");
		RoleVo roleClient = userService.getRoleByName("CLIENT");
		UserVo admin1 = new UserVo("admin1", "admin1", Arrays.asList(roleAdmin));
		UserVo admin2 = new UserVo("admin2", "admin2", Arrays.asList(roleAdmin));
		UserVo client1 = new UserVo("client1", "client1", Arrays.asList(roleClient));
		UserVo client2 = new UserVo("client2", "client2", Arrays.asList(roleClient));
		userService.save(admin1);
		userService.save(client1);
		userService.save(client2);
		userService.save(admin2);

		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { MediaType.APPLICATION_JSON }));
		// Request to return JSON format
		headers.setContentType(MediaType.APPLICATION_JSON);

		UserVo user = new UserVo();
		user.setUsername("admin1");
		user.setPassword("admin1");

		HttpEntity<UserVo> entity = new HttpEntity<UserVo>(user, headers);
```

```
                    // TokenVo tokenTest=new TokenVo()

                    ResponseEntity<TokenVo> response = this.restTemplate.exchange("http://localhost:" + port +
"/auth/signin",
                                        HttpMethod.POST, entity, TokenVo.class);
                    assertThat(response.getBody()).isNotNull();
                    TokenVo t = (TokenVo) response.getBody();
                    List<String> roles = t.getRoles();

                    //assertThat(roles).isNotEmpty();
                    assertThat(t).isNotNull();
                    assertThat(roles).isNotNull();
                    assertThat(roles.get(0)).isEqualTo("ADMIN");
          }
}
```

- Créer la classe suivante pour tester le contrôlleur EmpController :

```
package ma.formations.integration.presentation;

import static org.assertj.core.api.Assertions.assertThat;

import java.util.Arrays;

import org.junit.jupiter.api.Test;
import org.mockito.Spy;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.boot.web.server.LocalServerPort;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;

import ma.formations.domaine.EmpVo;
import ma.formations.domaine.UserVo;
import ma.formations.jwt.JwtUtils;
import ma.formations.service.IEmpService;
import ma.formations.service.IUserService;

@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
public class TestEmpController {
          @LocalServerPort
          private int port;

          @MockBean
          private IEmpService service;

          @Spy
          private IUserService userService;
```

```
	@Autowired
	private TestRestTemplate restTemplate;

	@Autowired
	private JwtUtils jwtUtils;

	@Test
	void testgetEmp() {
		//userService.save(new RoleVo("ADMIN"));
		UserVo user = new UserVo("admin1", "admin1", Arrays.asList(userService.getRoleByName("ADMIN")));
		//userService.save(user);

		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { MediaType.APPLICATION_JSON }));
		// Request to return JSON format
		headers.setContentType(MediaType.APPLICATION_JSON);

		String token = jwtUtils.generateJwtTokenWithString(user.getUsername());

		headers.add("Authorization", "Bearer " + token);

		HttpEntity<EmpVo[]> entity = new HttpEntity<EmpVo[]>(headers);

		// TokenVo tokenTest=new TokenVo()

		ResponseEntity<EmpVo[]> response = this.restTemplate.exchange("http://localhost:" + port +
"/employees",
				HttpMethod.GET, entity, EmpVo[].class);
		assertThat(response.getBody()).isNotNull();


	}
}
```

b. Tester la couche présentation avec MockMvc

- Créer la classe suivante pour tester l'authentification en utilisant MockMvc :

```
package ma.formations.integration.presentation;

import static org.mockito.Mockito.when;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import java.util.Arrays;

import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.test.web.servlet.MockMvc;

import ma.formations.domaine.UserVo;
import ma.formations.jwt.JwtUtils;
import ma.formations.service.IUserService;

@AutoConfigureMockMvc
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
class TestAuthenticationControllerWithMocking {

	@Autowired
	private MockMvc mockMvc;

	@MockBean
	AuthenticationManager authenticationManagerMock;

	@MockBean
	private IUserService userServiceMock;

	@MockBean
	private JwtUtils jwtUtilsMock;

	@Test
	public void shouldReturnHello() throws Exception {

		String tokenTest = "AAAA.BBBB.SSSS";
		UserVo userVoTest = new UserVo();
		userVoTest.setUsername("admin1");
		userVoTest.setPassword("admin1");

		Authentication authenticationResult = new
UsernamePasswordAuthenticationToken(userVoTest.getUsername(),
				userVoTest.getPassword(), Arrays.asList(new
SimpleGrantedAuthority("ADMIN")));

	when(authenticationManagerMock.authenticate(Mockito.any())).thenReturn(authenticationResult);
		when(jwtUtilsMock.generateJwtToken(authenticationResult)).thenReturn(tokenTest);
		mockMvc.perform(post("/auth/signin").contentType(MediaType.APPLICATION_JSON)
				.content("{ \"username\": \"admin1\", \"password\":
\"admin1\"}").accept(MediaType.APPLICATION_JSON))

	.andExpect(status().isOk()).andExpect(content().contentType(MediaType.APPLICATION_JSON))
				.andExpect(jsonPath("$.username").value("admin1"))
				.andExpect(jsonPath("$.jwttoken").value(tokenTest));
		}
}
```