Tests Spring

Dépendances pom.xml

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

        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!--
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <scope>test</scope>
        </dependency>
        <!--
https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
        <dependency>
            <groupId>io.github.bonigarcia</groupId>
            <artifactId>webdrivermanager</artifactId>
            <version>5.3.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

@WebMvcTest

Tester un composant (Controller, service...)

Controller

```
@Controller
public class HelloController {
  @Autowired
  private HelloService helloService;

  @GetMapping("/hello")
```

```
  public @ResponseBody String helloAction() {
    return helloService.getMessage();
  }

  @ModelAttribute("message")
  public String getMessage() {
    return helloService.getMessage();
  }

  @GetMapping("/hello/view")
  public String helloViewAction() {
    return "hello";
  }

  @GetMapping("/auth/hello")
  public @ResponseBody String authHelloAction() {
    return helloService.getAuthMessage();
  }

  @GetMapping("/hello/js/{msg}")
  public String helloWithJSAction(@PathVariable String msg) {
    return "helloJs";
  }
}
```

Test

```
@WebMvcTest(HelloController.class)
@ContextConfiguration(classes = {WebSecurityConfig.class,
SpringTestsApplication.class})
class HelloControllerTest {

  @MockBean
  private HelloService helloService;

  @Autowired
  private MockMvc mockMvc;

  @Test
  void helloShouldReturnBonjour() throws Exception {
    // Given
    when(helloService.getMessage()).thenReturn("Bonjour");
    // When
    ResultActions results =
this.mockMvc.perform(MockMvcRequestBuilders.get("/hello"));
    // Then
    results.andExpect(MockMvcResultMatchers.status().isOk())
        .andExpect(content().string(containsString("Bonjour")));
  }

  @Test
  void helloViewShouldReturnBonjour() throws Exception {
    // Given
```

```
    when(helloService.getMessage()).thenReturn("Bonjour");
    // When
    ResultActions results =
this.mockMvc.perform(MockMvcRequestBuilders.get("/hello/view"));
    // Then
    results.andExpect(view().name("hello")).andExpect(model().attribute("message",
"Bonjour"))
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andExpect(content().string(containsString("Bonjour")));
  }
}
```