**Junit Annotation**

**@Test** annotation specifies that method is the test method.

**@Test(timeout=1000)** annotation specifies that method will be failed if it takes longer than 1000 milliseconds (1 second).

**@BeforeClass** annotation specifies that method will be invoked only once, before starting all the tests.

**@Before** annotation specifies that method will be invoked before each test.

**@After** annotation specifies that method will be invoked after each test.

**@AfterClass** annotation specifies that method will be invoked only once, after finishing all the tests.

1. **void assertEquals(boolean expected,boolean actual)**: checks that two primitives/objects are equal. It is overloaded.
2. **void assertTrue(boolean condition)**: checks that a condition is true.
3. **void assertFalse(boolean condition)**: checks that a condition is false.
4. **void assertNull(Object obj)**: checks that object is null.
5. **void assertNotNull(Object obj)**: checks that object is not null.

@SpringBootTest & @AutoConfigureMockMvc

Or

@WebMvcTest

@AutoConfigureMockMvc

public class TestingWebApplicationTest {

@Autowired

private WebApplicationContext webApplicationContext;

private MockMvc mockMvc;

@Before

public void setup() {

mockMvc = MockMvcBuilders.webAppContextSetup(webApplicationContext).build();

}

@Test

public void testEmployee() throws Exception {

mockMvc.perform(get("/employee")).andExpect(status().isOk()).

andExpect(content().contentType("application/json;charset=UTF-8"))

.andExpect(jsonPath("$.name").value("emp1")).andExpect(jsonPath("$.designation").value("manager")).andExpect(jsonPath("$.empId").value("1")).andExpect(jsonPath("$.salary").value(3000));

}

}

**@MockBean** We use @MockBean to create and inject a mock for the GreetingService (if you do not do so, the application context cannot start), and we set its expectations using Mockito.

**@LocalServerPort: Note the use of webEnvironment=RANDOM\_PORT to start the server with a random port (useful to avoid conflicts in test environments) and the injection of the port with @LocalServerPort.**

@SpringBootTest

public class SmokeTest {

@Autowired

private HomeController controller;

@Test

public void contextLoads() throws Exception {

assertThat(controller).isNotNull();

}

}

@SpringBootTest(webEnvironment = WebEnvironment.RANDOM\_PORT)

public class HttpRequestTest {

@LocalServerPort

private int port;

@Autowired

private TestRestTemplate restTemplate;

@Test

public void greetingShouldReturnDefaultMessage() throws Exception {

assertThat(this.restTemplate.getForObject("http://localhost:" + port + "/",

String.class)).contains("Hello, World");

}

}

**Mokito**

//@InjectMocks annotation is used to create and inject the mock object

@InjectMocks

MathApplication mathApplication = new MathApplication();

@Test

public void testAdd(){

//add the behavior of calc service to add two numbers

when(calcService.add(10.0,20.0)).thenReturn(30.00);

//test the add functionality

Assert.assertEquals(mathApplication.add(10.0, 20.0),30.0,0);

}