Bài 16
Automated Testing

Module: WBD

Mục tiêu

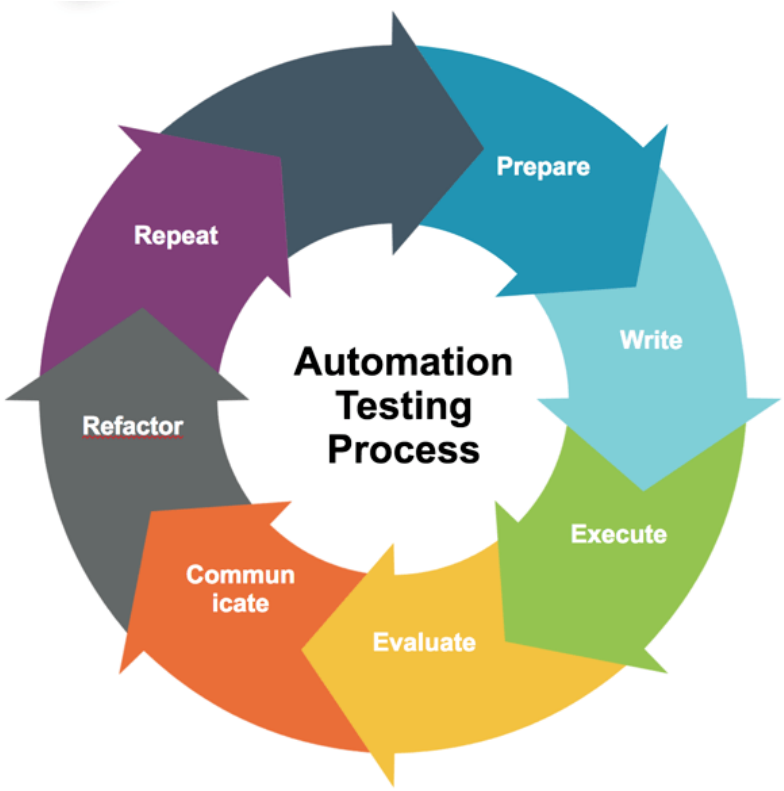
- Trình bày được khái niệm Automated Testing
- Triển khai được Automated Testing trong Spring MVC

Automated Testing

Automated Testing là gì

Automated Testing là gì

Kiểm thử tự động là một quá trình xử lý tự động các bước thực hiện một test case. Kiểm thử tự động được thực hiện bởi phần mềm kiểm thử tự động - hay còn gọi là Automation Testing Tool.



Một số phần mềm kiểm thử tự động

- Quick Test Profressional - (HP)
- Selenium
- Test Architect - (LogiGear)
- Visual Studio CodedUI Testing
- TestComplete (SmartBear)
- SOAPUI – Web Services Testing (SmartBear)
- Ranorex

Ưu điểm của Automated Testing

- Độ tin cậy cao
- Khả năng lặp
- Tốc độ cao
- Chi phí thấp
- Khả năng tái sử dụng

Nhược điểm của Automated Testing

- Khó mở rộng, khó bảo trì
- Khả năng bao phủ thấp
- Vấn đề công cụ và nhân lực

Triển khai Automated Testing trong Spring MVC

Cấu hình thưviện

- Thêm dependency vào file build.gralde

```
testCompile group: 'org.junit.jupiter', name: 'junit-jupiter-engine', version: '5.7.0'
testCompile group: 'org.junit.platform', name: 'junit-platform-commons', version: '1.7.0'
testCompile group: 'org.junit.platform', name: 'junit-platform-launcher', version: '1.7.0'
testCompile group: 'org.springframework', name: 'spring-test', version: '5.3.2'
testCompile group: 'com.github.sbrannen', name: 'spring-test-junit5', version: '1.2.0'
testCompile group: 'org.mockito', name: 'mockito-all', version: '1.10.19'
```

- Bổ sung đoạn script vào file build.gradle để chạy test bằng junit platform

```
test {
    useJUnitPlatform()

    testLogging {
        events "PASSED", "STARTED", "FAILED", "SKIPPED", "STANDARD_OUT", "STANDARD_ERROR"
    }
    afterTest { desc, result ->
        println "Testing ${desc.name} [${desc.className}]: ${result.resultType}"
    }
    reports {
        html.enabled = true
    }
}
```

Triển khai Spring Test

- Sử dụng @SpringJUnitJupiterConfig để nạp các đối tượng phụ thuộc khác, tự định nghĩa, ví dụ:

```
@Configuration
@ComponentScan("cg.wbd.grandemonstration")
public class CustomerControllerTestConfig {
    @Bean
    public DataSource dataSource() {
        return new EmbeddedDatabaseBuilder()
                .setType(EmbeddedDatabaseType.H2)
                .setName("cms")
                .build();
    }
}
```

- Khi sử dụng junit flatform, JUnit Framework đã có khả năng cấu hình compile, build và thực thi kiểm thử mà không cần chỉ định gì đặc thù.

```
@WebAppConfiguration
@SpringJUnitJupiterConfig(CustomerControllerTestConfig.class)
public class CustomerControllerTest {
```

Triển khai Spring Test

- Giả lập các đối tượng phụ thuộc:

```
private CustomerService customerService = Mockito.mock(CustomerServiceImplWithSpringData.class);
private ProvinceService provinceService = Mockito.mock(ProvinceServiceImplWithSpringData.class);
private MockMvc mockMvc;
@InjectMocks
private CustomerController customerController;
@BeforeEach
void setUp() {
    MockitoAnnotations.initMocks(this);
    mockMvc = MockMvcBuilders
            .standaloneSetup(customerController)
            .setCustomArgumentResolvers(new PageableHandlerMethodArgumentResolver())
            .build();
```

Kiểm thử tên View

```
@Test
void customerBrowseControlling() throws Exception {
    mockMvc
            .perform(get("/customers"))
            .andExpect(status().is(200))
            .andExpect(view().name("customers/browse"));
}
```

- perform(get(…)) sẽ giả lập gửi một request tới url chỉ định
- andDo print giúp in request và response tới luồng out, phục vụ debug
- expect view name xác nhận xem có đúng là view chỉ định được sử dụng không

Kiểm thử response body

```
@Test
void customersListPageIsExists() throws Exception {
    mockMvc
            .perform(get("/customers"))
            .andExpect(status().is(200));
}
```

- expect status isOk xác nhận rằng HTTP code của response là 200

Tổng kết

- Kiểm thử tự động là một quá trình xử lý tự động các bước thực hiện một test case
- Triển khai kiểm thử ứng dụng Spring MVC bằng JUnit Platform

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: Spring Boot