TEST & CICD

Andy

1

Work Environment

- Local -> DEV -> QA -> Pre-Prod -> PROD
- Git -> Bitbucket/Github
- Agile – Sprint (Planing -> Grooming -> Daily Standup -> Retro)
- Team Organization (PO, Business, Dev, QA)
- JIRA (Tickets – new story/bug fixing, Points-based)

2

General Test in industry

Environment	Test Types	Tools
Local	Peer Test Unit Test	Human Junit, Mockito, PowerMock etc
Dev	Unit Test Automation Test	Junit etc Selenium, TestNG, Cucumber etc
QA	Manual Automation Regression	Human, HP Quality Center Selenium, TestNG, Cucumber etc Automation test
Pre-Prod	Performance/Load Test	JMeter, Gatling
Prod	AB Test Alpha/Beta test (Prod-Log Retrieve)	User

3

JUnit Test

```
@Service
public class UserDetailsServiceImpl{

    @Autowired
    UserRepository userRepository;

    public User getUserByEmail(String email) throws SQLException{
        User user = userRepository.getUserByEmail(email);
        if(user == null){
            throw new SQLException("cannot find user");
        }
        return user;
    }

    public Set<User> getUsersList(){
        return userRepository.getUsers();
    }
}
```

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class UserDetailsServiceImplTest {

    @Autowired
    UserDetailsServiceImpl service;

    @Before
    public void beforeTest(){
        System.out.println("test service class: started");
    }

    @After
    public void afterTest(){
        System.out.println("test service class: completed");
    }

    @Test
    public void testGetUsersList(){
        Set<User> users = service.getUsersList();
        Assert.assertEquals(2, users.size());
    }

    @Test(expected = SQLException.class)
    public void testGetUserByEmail() throws SQLException{
        System.out.println("test GetUserByEmail() started");
        String email = "C@gmail.com";
        User user = service.getUserByEmail(email);
        Assert.assertEquals("USER", user.getRole());
        Assert.assertEquals(1, user.getId());
        System.out.println("test GetUserByEmail() completed");
    }
}
```

4

Mock: Database

```
public User getUserById(int id){
    return userRepository.getUserById(id);
}
```

```
@MockBean
UserRepository repository;

//@InjectMocks
@Autowired
UserDetailsServiceImpl service;

@Test
public void testGetUserById(){
    System.out.println("mockito test GetUserById() started");
    int id = 3;
    User mockUser = new User( id: 3,  email: "mock@test.com",  password: "mocktest",  role: "NA");
    when(repository.getUserById(id)).thenReturn(mockUser);
    User user = service.getUserById(id);
    Assert.assertEquals( expected: "mock@test.com", user.getEmail());
    System.out.println("mockito test GetUserById() completed");
}
```

5

Mock: REST API

```
@PostMapping("/register")
public String registerUser(@ModelAttribute User user){
    System.out.println(user);
    service.saveUser(user);
    return "loginPage";
}
```

```
@Test
public void testRegisterUserMockMVC() throws Exception{

    System.out.println("MockMVC test registerUser started");

    User mockUser = new User( id: 3,  email: "mock@test.com",  password: "mocktest",  role: "NA");
    doNothing().when(service).saveUser(mockUser); //doNothing is for void method

    mockMvc.perform(
                MockMvcRequestBuilders
                    .post( urlTemplate: "/register")
                    .flashAttr( name: "user", mockUser)   //for modelAttribute
                    .contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON)
            ).andExpect(status().isOk()) //redirect code : 302
            .andExpect(MockMvcResultMatchers.view().name( expectedViewName: "loginPage"));

    System.out.println("MockMVC test registerUser completed");
}
```

6

Mock: RestTemplate

```
public User getUser(int id) {
    String url = "http://localhost:2020/customer-service/customer/" + id;
    User user = this.restTemplate.getForObject(url, User.class);
    return user;
}
```

```
//Mockito - for RestTemplate
@Test
public void testGetRemoteUserInfo(){
    System.out.println("mockito test GetRemoteUserInfo() started");
    int id = 3;
    User mockUser = new User( id: 3,  email: "mock@test.com",  password: "mocktest",  role: "NA");

    String url = "http://localhost:2020/customer-service/customer/" + id;
    when(restTemplate.getForObject(url, User.class)).thenReturn(mockUser);

    User user = service.getRemoteUserInfo(id);
    Assert.assertEquals( expected: "mock@test.com", user.getEmail());

    System.out.println("mockito test GetRemoteUserInfo() completed");
}
```

7

Test Suite - Automation

```
@RunWith(Suite.class)
@Suite.SuiteClasses({
        MainControllerMockTest.class,
        UserDetailsServiceImplTest.class,
        UserDetailsServiceImplMockTest.class
})
public class CompositeApplicationTests {

    @Test
    public void contextLoads() {
    }

}
```

8

Automation whole application from front end

- Selenium
  - WebDriver – a driver to help grab web elements
  - Page Objects – a java class to simulate a web page
- TestNG (Data-Driven Development)
- Cucumber

9

Selenium

user-behavior simulation

```
@Test
public void testLogin(){
    driver.get("http://localhost:8080");

    WelcomePage welcomePage = new WelcomePage(driver);
    welcomePage.clickLoginLink();

    LoginPage loginPage = new LoginPage(driver);
    loginPage.login();

    UserInfoPage userInfoPage = new UserInfoPage(driver);

    Assert.assertTrue(driver.getPageSource().contains("User Info page"));
}
```

```
public class WelcomePage{

    @FindBy(id = "login")
    private WebElement loginButton;

    @FindBy(xpath = "//div[@class='col-sm-12']//h2")
    private WebElement welcomeMessage;

    public WelcomePage(WebDriver driver) {
        WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds: 5);
        wait.until(ExpectedConditions.presenceOfElementLocated(By.id("login")));
        PageFactory.initElements(driver, page: this);
    }

    //methods
    public void clickLoginLink(){
        loginButton.click();
    }
}
```

10

Cucumber

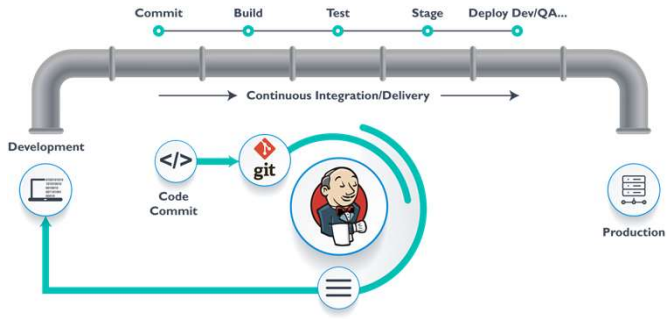
- TDD - Test Driven Development
  - Requirement -> Test case -> implementation -> code rephrase
- BDD - Behavior-Driven Development
  - Requirement being part of test case
  - No technical people
  - CUCUMBER
    - Gherkin language
    - FEATURE File
    - STEP definition
    - Test Case

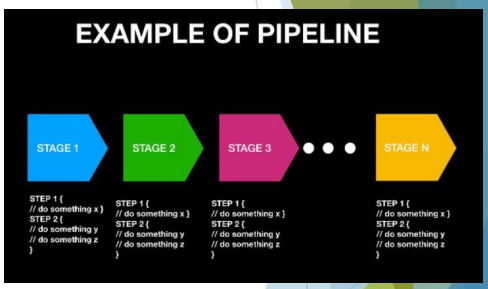
```
Feature: Login functionalities

  Scenario: Login with correct credentials
    Given web browser loaded the welcome page
    When the user click login link in welcome page
    Then login page should be displayed
```

```
@Given("web browser loaded the welcome page")
public void webb_browser_is_open(){
    driver.get("http://localhost:8080");
}


@When("^the user click login link in welcome page$") //
public void click_login_link_in_welcome_page(){
    WelcomePage welcomePage = new WelcomePage(driver);
    welcomePage.clickLoginLink();
}

@Then("^login page should be displayed$")
public void login_page_displayed(){
    String source = driver.getPageSource();
    Assert.assertTrue(source.contains("Email:"));
}
```

11

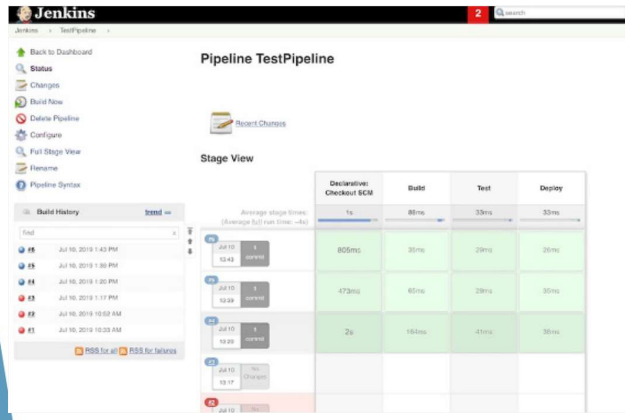
CICD





12

Jenkins Pipeline



READ:

https://opensource.com/article/19/9/intro-building-cicd-pipelines-jenkins

```
//Jenkinsfile (Declarative Pipeline)
pipeline {
    agent { docker { image 'maven:3.3.3' } }
    stages {
        stage("Build") {
            steps {
                echo "Building ${env.Build_ID}"
                javac HelloWorld.java
                mvn clean package ./HelloWorld
            }
        }
        stage('Deploy') {
            steps {
                sh './deploy-script' //run deploy script
            }
        }
        stage('Test') {
            steps {
                mvn test HelloWordTest
            }
        }
    }
    post {
        always {
            echo 'Generate Report File'
            junit 'build/reports/reports.xml'
        }
        success {
            echo 'This will run only if successful'
        }
        failure {
            mail to: 'team@example.com',
             subject: "Failed Pipeline: ${currentBuild.fullDisplayName}",
             body: "Something is wrong with ${env.BUILD_URL}"
        }
    }
}
```

13