BDD - Behaviour Driven Development

Das Hauptziel von BDD ist, dass alle an einem Projekt beteiligten Personen (Business Analysts, Quality Assurance, Developers, Stakeholder) verstehen "was" gemeint ist.

Collaboration between Business and Technical Team

- BDD is all about collaboration betweem teams (tech & business)
- to build a common understanding on the behaviour of the application
- Generate a common documentation that can be understood by all teams & stakeholders

BDD Developmemt Process

1. Stakeholder & product owner talk about business needs
2. Product owner, Developer and Tester collaborate around requirements ("Three Amigos")
3. Agreed upon requirements defined as English-formatted Scenarios (Gherkin)
4. Dev team uses Scenarios for manual Testcases
5. Tester team uses Scenarios for automated tests
6. Automated Test Reports generated against Features/Scenarios
7. Release

For any new enhancement, feature, change --> the user story is created --> an informal, general explanation of a software feature written from the perspective of the end user

1. it is discussed among the teams
2. conversation & discussion on how the system should behave
3. examples are created, discussed, agreed and approved
4. examples are documented in a way that can be developed & tested with automation
5. coding phase --> implement behaviour as per the development examples

3 Phases

Before the 3 Phases
Business Analyst takes requirements from the stakeholder

Discovery Phase
starts with the "Three Amigos"-Meeting (Business Analyst representing the Business Owner, Dev, Tester)
explore & discuss how the system should behave --> User Story
= agreed behaviour of the system

Formulation Phase
create concrete exmples that can be automated, Gherkin translated
= documented examples (acceptance test)

Automation Phase
coding of the documented examples to implement the agreed behaviour
create automation tests taking one example at a time to guide development

test execution, bug fixes, retesting, release
= code implementation & automation tests

Sprint
A sprint is a short, time-boxed period where the team works to complete a set amount of work

Tools for BDD
- Cucumber
- J Behave
- Behat

BDD vs. ATDD
BDD desccribes overall behaviour of the system, customer focused
ATDD defines tests which act as requirements for the system, development focused

You need 2 Files – Features and Step Definition to execute a Cucmber test scenario
Features file contain high level description of the Test Scenario in simple language
Steps Definition file contains the actual code to execute the Test Scenario in the Features file.

Feature File:

A feature file is a plain text file that contains high-level, user-centric descriptions of the behavior and expected outcomes of the system under development. It is written in a specific format called Gherkin, which is a simple, natural language syntax that makes it easy for non-technical stakeholders to understand the requirements and provide feedback. Feature files are a core part of behavior-driven development (BDD), and they serve as the foundation for the automated test cases that are implemented in the step definitions. It is also **the source of truth** for what the system should do and how it should behave, and it is the reference point for developers, testers, and stakeholders to understand the requirements and ensure that the implementation meets the desired behavior.

All BDD Frameworks are written in "Gherkin".

Gherkin
Hat eine line-oriented (zeilenorientiert) desgin Syntax ähnlich zu Phyton.
Struktur ist whitespace/tab sensitive, liest alle non-empty und non-comment Zeilen
Aufgeteilt in Feature, Scenarios, Steps

Feature
- Feature: Keyword
- Feature Name
- Description

Feature Example Defintion

Feature Withdraw Money
Scenario Scenario 1
Given preconditions
WHENactions
THEN results

Jedes Feature besteht aus einer Sammlung Scenarios. Ein Senario beschreibt mehrere Events eines Features.

Scenarios

Scenario A user withdraws money from an ATM
Given <Name> has a valid card
AND their balance is <OriginalBalance>
WHENhe inserts card
AND withdraws <WithdrawalAmount>
THEN ATM returns <WithdrawalAmount>
AND their Account balance is <NewBalance>

Steps

Keywords

- Given (precondition)
- WHEN (actions)
- THEN (results)
- AND
- BUT

Selenium WebDriver testing framework

LogIn_Test.fetaure

Feature: Login Action

Scenario: Successful Login with Valid Credentials
Given User is on Home Page
When User Navigate to LogIn Page
And User enters UserName and Password
Then Message displayed Login Successfully

Scenario: Successful LogOut
When User LogOut from the Application
Then Message displayed LogOut Successfully

Test_Steps.java

```
package stepDefinition;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;

public class Test_Steps {
	public static WebDriver driver;
	@Given("^User is on Home Page$")
	public void user_is_on_Home_Page() throws Throwable {
		driver = new FirefoxDriver();
		driver.manage().timeouts().implicitlyWait(10,
	TimeUnit.SECONDS);
	    driver.get("https://www.store.demoqa.com");
	}

	@When("^User Navigate to LogIn Page$")
	public void user_Navigate_to_LogIn_Page() throws Throwable {
		driver.findElement(By.xpath(".//*[@id='account']/a")).click();
	}

	@When("^User enters UserName and Password$")
	public void user_enters_UserName_and_Password() throws Throwable {
		driver.findElement(By.id("log")).sendKeys("testuser_1");
	    driver.findElement(By.id("pwd")).sendKeys("Test@123");
	    driver.findElement(By.id("login")).click();
	}

	@Then("^Message displayed Login Successfully$")
	public void message_displayed_Login_Successfully() throws Throwable {
		System.out.println("Login Successfully");
	}
```

```
	@When("^User LogOut from the Application$")
	public void user_LogOut_from_the_Application() throws Throwable {
		driver.findElement (By.xpath(".//*[@id='account_logout']/a")).click();
	}

	@Then("^Message displayed LogOut Successfully$")
	public void message_displayed_LogOut_Successfully() throws Throwable {
		System.out.println("LogOut Successfully");
	}
}
```

TestRunner.java

```
package cucumberTest;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
		features = "Feature"
		,glue={"stepDefinition"}
		)

public class TestRunner {}
```