Cucumber - Understanding BDD

✅ What is BDD (Behavior-Driven Development)? — For Beginners

BDD (Behavior-Driven Development) is a **software development** approach that combines the **technical aspects of TDD (Test-Driven Development)** with **business-oriented language**.

🔍 In Simple Words:

BDD allows you to write test cases in **plain English** so **non-technical stakeholders** (like product owners or testers) can read and understand them.

💡 Example Use Case:

Instead of writing:

```java
assertEquals("Login Successful", loginPage.getMessage());
```

You write a **readable scenario** like:

```pgsql
Given user is on login page
When user enters valid credentials
Then user should be navigated to the dashboard
```

🧩 BDD Key Components:

Term	Meaning
Feature	A high-level description of the functionality
Scenario	A concrete example illustrating behavior
Given	Sets the context (preconditions)
When	The action taken by the user
Then	The expected outcome or result

🛠️ BDD Tools in Java:

The most popular Java-based BDD tool is **Cucumber**.

✅ Sample BDD Code with Cucumber + Selenium

📁 Folder Structure:

```
pgsql

project/
├── features/
│   └── login.feature
├── stepdefinitions/
│   └── LoginSteps.java
├── testrunner/
│   └── TestRunner.java
```

📄 1. `login.feature` (Gherkin Syntax)

```
gherkin

Feature: Login Functionality

  Scenario: Successful login with valid credentials
    Given user is on the login page
    When user enters username "admin" and password "admin123"
    Then user should be navigated to the dashboard
```

📄 2. `LoginSteps.java`

```
java

package stepdefinitions;

import io.cucumber.java.en.*;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;

public class LoginSteps {
    WebDriver driver;

    @Given("user is on the login page")
    public void user_is_on_login_page() {
        driver = new ChromeDriver();
        driver.get("https://example.com/login");
    }

    @When("user enters username {string} and password {string}")
    public void user_enters_credentials(String username, String password) {
        driver.findElement(By.id("username")).sendKeys(username);
        driver.findElement(By.id("password")).sendKeys(password);
        driver.findElement(By.id("loginButton")).click();
    }

    @Then("user should be navigated to the dashboard")
    public void user_should_be_on_dashboard() {
        String dashboardTitle = driver.getTitle();
        if (!dashboardTitle.contains("Dashboard")) {
            throw new AssertionError("User is not on Dashboard!");
        }
        driver.quit();
    }
}
```

📄 3. `TestRunner.java`

```java
package testrunner;

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

@RunWith(Cucumber.class)
@CucumberOptions(
  features = "features",
  glue = "stepdefinitions",
  plugin = {"pretty", "html:target/cucumber-reports.html"}
)
public class TestRunner {
}
```

✅ What You Achieve with BDD:

- **Readable** and **maintainable** test cases
- **Bridges the gap** between technical and non-technical teams
- Encourages **collaboration**

🚀 Summary:

Concept	Explanation
BDD	Writing test cases in natural language (Gherkin)
Tools	Cucumber, JBehave
Benefit	Better collaboration and clear expectations

Would you like a **mini-project** with BDD + Selenium + Maven to practice?

Here's a beginner-friendly comparison of **BDD vs TDD**, with clear explanations and examples.

🆚 BDD vs TDD — For Beginners

Feature	TDD (Test-Driven Development)	BDD (Behavior-Driven Development)
🧠 **Focus**	Testing the **internal logic** of the code (unit tests)	Testing the **behavior** of the system from the user's view
🗣️ **Language**	Code-level language (e.g., Java, Python)	Natural language (e.g., English using Gherkin syntax)
👥 **Audience**	Developers	Developers, testers, and non-technical stakeholders
📄 **Test Format**	Assertions in test methods	Scenarios written in plain English (Given/When/Then)
🛠️ **Tools**	JUnit, TestNG, NUnit, PyTest	Cucumber, SpecFlow, Behave
✅ **Example Test**	`assertEquals(5, calculator.add(2, 3))`	`Given the user inputs 2 and 3, When added, Then result is 5`
🤝 **Collaboration**	Mainly among developers	Encourages cross-functional collaboration (Dev + QA + BA)

🔧 TDD Example (in Java using JUnit)

```
java

@Test
public void testAddition() {
    Calculator calc = new Calculator();
    int result = calc.add(2, 3);
    assertEquals(5, result);
}
```

🌍 BDD Example (Gherkin + Java using Cucumber)

Feature File:

```
gherkin

Scenario: Adding two numbers
  Given user has entered 2 and 3
  When user adds the numbers
  Then the result should be 5
```

Step Definition (Java):

```
java

@Given("user has entered {int} and {int}")
public void enter_numbers(int num1, int num2) {
    calculator.setInput(num1, num2);
}

@When("user adds the numbers")
public void add_numbers() {
    calculator.add();
}

@Then("the result should be {int}")
public void check_result(int expected) {
    assertEquals(expected, calculator.getResult());
}
```

🎯 When to Use What?

Situation	Use
Testing low-level code logic (e.g., methods)	✅ TDD
Testing behavior & workflow (e.g., login flow)	✅ BDD
Need stakeholder collaboration	✅ BDD
Focused developer-only unit testing	✅ TDD

📝 Summary

Criteria	TDD	BDD
Syntax	Programming code	Natural language (Gherkin)
Focus Area	Internal code (units)	External behavior (features)
Collaboration	Developers	Dev + QA + Business
Output	Unit tests	Feature scenarios

Let me know if you'd like a combined **BDD + TDD real-world mini project example** (like login or calculator).

Great question! Here's a simple and beginner-friendly explanation:

✅ Why BDD is Preferred in Automation?

🔍 What is BDD in Automation?

BDD (Behavior-Driven Development) in automation means writing your test scenarios in **plain English** (using tools like **Cucumber**) so that **everyone on the team** — including **non-technical people** — can understand what's being tested.

🧠 Why Is BDD Preferred in Automation? (Especially for beginners)

Here are **7 simple reasons**:

---

1. ✅ Readable and Understandable Test Cases

- Written in plain English (Gherkin syntax)
- Example:

```gherkin

Scenario: Successful login
  Given user is on login page
  When user enters valid credentials
  Then user should be redirected to dashboard
```

➡️ Even **non-programmers** (like testers or product owners) can read and understand it.

---

2. 👥 Better Collaboration

- Developers, QA, and Business Analysts work together
- Everyone speaks the **same language**
- Helps prevent miscommunication

---

3. 🔄 Reusability of Code

- Step definitions (like `Given`, `When`, `Then`) can be reused across multiple tests
- Less duplication, easy to maintain

---

4. 🎯 Focuses on Behavior, Not Just Code

- BDD tests the **user behavior** (real-world scenarios), not just functions
- Helps ensure the system does **what the user actually wants**

5. 🔪 Integrates Easily with Automation Tools

- Works well with:
  - **Selenium** (for UI testing)
  - **Appium** (for mobile apps)
  - **Rest Assured** (for API testing)
- Tools like **Cucumber** connect English scenarios with Java/Python code

---

6. 📜 Living Documentation

- Feature files act as **documentation + test cases**
- Always **up-to-date** and understandable
- Helps during knowledge transfer

---

7. 🚀 Faster Development and Fewer Bugs

- Scenarios are defined **before coding**
- Developers write code to pass the expected behavior
- Reduces bugs and rework

🧩 Real-Life Example

🚫 Traditional Automation Test:

```
java

@Test
public void testLogin() {
  driver.findElement(By.id("username")).sendKeys("admin");
  driver.findElement(By.id("password")).sendKeys("admin123");
  driver.findElement(By.id("loginBtn")).click();
  assertEquals("Dashboard", driver.getTitle());
}
```

✅ BDD Test:

```
gherkin

Scenario: Successful login
  Given user is on login page
  When user logs in with username "admin" and password "admin123"
  Then user should see the dashboard
```

➡️ Anyone can **understand the scenario**, not just developers.

📝 Summary Table

Advantage	Why it matters
✅ Readable Tests	Easy for all team members to understand
🤝 Better Collaboration	Dev + QA + BA work together
🔄 Reusable Steps	Saves time and avoids duplication
🧭 Behavior Focus	Tests real user scenarios
📘 Living Docs	Scenarios act as both tests and docs

Would you like a **mini BDD automation framework** example using Selenium + Cucumber to practice?

Absolutely! Let's walk through a **real-world BDD example** using a simple, familiar scenario:

🎯 Real-World Example: Login Feature for a Web Application

We'll use **BDD with Cucumber** for testing a **login functionality** using **Selenium (Java)**.

💡 Scenario

> A user visits a login page.
> If they enter correct credentials, they should be taken to the dashboard.

🧱 Folder Structure (Simple)

```pgsql
project/
├── features/
│   └── login.feature
├── stepdefinitions/
│   └── LoginSteps.java
├── testrunner/
│   └── TestRunner.java
├── pages/
│   └── LoginPage.java
```

1️⃣ `login.feature` — Plain English Scenario (Gherkin syntax)

```gherkin
Feature: Login Functionality

  Scenario: Successful login with valid credentials
    Given user is on the login page
    When user enters username "admin" and password "admin123"
    Then user should be redirected to the dashboard
```

2️⃣ `LoginPage.java` — Page Object (Selenium)

```java
package pages;

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

public class LoginPage {
    WebDriver driver;

    // Locators
    By usernameField = By.id("username");
    By passwordField = By.id("password");
    By loginButton = By.id("loginBtn");
```

```
    public LoginPage(WebDriver driver) {
        this.driver = driver;
    }

    public void enterUsername(String username) {
        driver.findElement(usernameField).sendKeys(username);
    }

    public void enterPassword(String password) {
        driver.findElement(passwordField).sendKeys(password);
    }

    public void clickLogin() {
        driver.findElement(loginButton).click();
    }

    public String getPageTitle() {
        return driver.getTitle();
    }
}
```

3️⃣ `LoginSteps.java` — Step Definitions (Glue code)

```
java

package stepdefinitions;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.cucumber.java.en.*;
import pages.LoginPage;

import static org.junit.Assert.*;

public class LoginSteps {
    WebDriver driver;
    LoginPage loginPage;

    @Given("user is on the login page")
    public void user_is_on_login_page() {
        driver = new ChromeDriver(); // Make sure chromedriver is in path
        driver.get("https://example.com/login"); // Replace with real URL
        loginPage = new LoginPage(driver);
    }

    @When("user enters username {string} and password {string}")
    public void user_enters_credentials(String username, String password) {
        loginPage.enterUsername(username);
        loginPage.enterPassword(password);
        loginPage.clickLogin();
    }

    @Then("user should be redirected to the dashboard")
    public void user_should_see_dashboard() {
        String expectedTitle = "Dashboard";
        String actualTitle = loginPage.getPageTitle();
        assertTrue("Login failed!", actualTitle.contains(expectedTitle));
        driver.quit();
    }
}
```

4️⃣ `TestRunner.java` — To Run All Scenarios

```java
package testrunner;

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

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "features",
    glue = "stepdefinitions",
    plugin = {"pretty", "html:target/cucumber-report.html"}
)
public class TestRunner {
}
```

✅ What This Shows

- **`login.feature`**: The behavior is written in **plain English**.
- **`LoginSteps.java`**: The steps behind each sentence are mapped to **code**.
- **`LoginPage.java`**: Selenium code is separated using **Page Object Model**.
- **`TestRunner.java`**: Executes the tests and generates a report.

🔧 Prerequisites

Make sure your Maven **`pom.xml`** has dependencies for:

```xml
<dependencies>
  <dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.14.0</version>
  </dependency>
  <dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>7.14.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.21.0</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
  </dependency>
</dependencies>
```

✅ Summary for Beginners

Part	Purpose
`login.feature`	Human-readable test case
`StepDefinitions`	Connects natural language to code
`LoginPage`	Contains actual Selenium automation
`TestRunner`	Executes the feature files

Would you like this in a ready-to-run **Maven GitHub project**, or want a version using **API testing with Rest Assured + BDD** instead?