# Selenium

## WebDriver.findElement vs WebElement.findElement

WebDriver.findElement(By by): Searches from the document root.

WebElement.findElement(By by): Searches within the context of that element.

WebElement form = driver.findElement(By.id("loginForm"));

WebElement username = form.findElement(By.name("username"));

## Supported Locators

By.id, By.name, By.className, By.tagName, By.linkText, By.partialLinkText, By.cssSelector, By.xpath

import static org.openqa.selenium.support.locators.RelativeLocator.with;

WebElement loginBtn = driver.findElement(with(By.tagName("button")).below(pwd));

## Soft vs Hard Assertions

Assert.assertEquals(actual, expected); // Hard

SoftAssert soft = new SoftAssert();

soft.assertTrue(title.contains("Dashboard"));

soft.assertAll();

# Java

## StringBuffer vs StringBuilder

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World");

## this and super Keywords

class B extends A {

B(String name){ super(name); this.name = name; }

}

## Array vs ArrayList

int[] arr = new int[3];

List<Integer> list = new ArrayList<>();

list.add(10);

## Abstract Class vs Interface

Abstract class can have state and constructors. Interface is for capabilities.

## Streams Example

List<Employee> emps = List.of(

new Employee("A", "QA", 9.0),

new Employee("B", "Dev", 15.0)

);

List<String> topQA = emps.stream()

.filter(e -> e.getDept().equals("QA"))

.map(Employee::getName)

.toList();

# API

## PUT vs POST

POST: Create new resource. Not idempotent.

PUT: Replace resource. Idempotent.

## Common HTTP Methods

GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS

## JSON Serialization & Deserialization

given().body(new User("sam", "QA"))

.when().post("/users")

.then().statusCode(201);

User user = given().get("/users/1").as(User.class);

## JSON Parsing Techniques

String name = response.path("name");

List<User> users = response.as(new TypeRef<List<User>>() {});

## Deserialization using POJO

public class User {

private int id;

private String name;

private String role;

public User() {}

public User(String name, String role) { this.name = name; this.role = role; }

}

User user = response.as(User.class);

# Jenkins

## What is Jenkins?

Open-source automation server for CI/CD.

## CI vs CD vs Continuous Deployment

CI: Merge and test code.

CD: Keep code deployable.

Continuous Deployment: Auto deploy every green build.

## Jenkins Pipeline

Pipeline-as-code using Jenkinsfile.

## Scripted vs Declarative Pipelines

Declarative: Simpler syntax. Scripted: Groovy-based, flexible.

## Sample Declarative Pipeline

pipeline {

agent any

stages {

stage('Build') {

steps {

sh 'mvn clean install'

}

}

}

}

**package** basePackage;

**import** java.util.Arrays;

**import** java.util.HashMap;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Set;

**public** **class** CollectionLoops {

**public** **static** **void** main(String[] args) {

//## 🔁 1. \*\*Looping through a List\*\*

List<String> fruits = Arrays.*asList*("Apple", "Banana", "Mango");

// Using for loop

**for** (**int** i = 0; i < fruits.size(); i++) {

System.***out***.println(fruits.get(i));

}

// Using enhanced for loop

**for** (String fruit : fruits) {

System.***out***.println(fruit);

}

// Using forEach with lambda

fruits.forEach(fruit -> System.***out***.println(fruit));

//## 🔁 2. \*\*Looping through a Set\*\*

Set<String> cities = **new** HashSet<>(Arrays.*asList*("Chennai", "Bangalore", "Hyderabad"));

// Enhanced for loop

**for** (String city : cities) {

System.***out***.println(city);

}

// Using iterator

Iterator<String> iterator = cities.iterator();

**while** (iterator.hasNext()) {

System.***out***.println(iterator.next());

}

//## 🔁 3. \*\*Looping through a Map\*\*

Map<String, Integer> scores = **new** HashMap<>();

scores.put("Math", 90);

scores.put("Science", 85);

scores.put("English", 88);

// Loop through keys

**for** (String subject : scores.keySet()) {

System.***out***.println(subject + ": " + scores.get(subject));

}

// Loop through entries

**for** (Map.Entry<String, Integer> entry : scores.entrySet()) {

System.***out***.println(entry.getKey() + " => " + entry.getValue());

}

// Using forEach with lambda

scores.forEach((subject, score) -> System.***out***.println(subject + ": " + score));

}

}

//Deserialize the response into PojoClasses

PojoClasses course = *given*().queryParam("access\_token", accessToken)

.when().get("https://rahulshettyacademy.com/oauthapi/getCourseDetails")

.as(PojoClasses.**class**);

//Print the values

System.***out***.println("LinkedIn: "+course.getlinkedIn());

System.***out***.println("Services: "+course.getServices());

System.***out***.println("Instructor: "+course.getInstructor());

System.***out***.println("Expertise: "+course.getExpertise());

System.***out***.println("Courses: "+course.getCourses());