

# Selenium TestNG Maven Automation Framework

This repository is a ready-to-run **Selenium 4 + TestNG + Maven** automation framework optimized to avoid Google reCAPTCHA by using **Bing**. It includes parallel execution, POM structure, utilities, reporting, screenshots, TestNG listeners, and CI hints (Jenkinsfile).

---

## Project Structure

```
selenium-framework/  
├ pom.xml  
├ README.md  
├ testng.xml  
├ Jenkinsfile  
├ src/  
│   ├── main/  
│   │   └── java/  
│   │       ├── com.example.framework/  
│   │       │   ├── base/BrowserFactory.java  
│   │       │   ├── base/BaseTest.java  
│   │       │   ├── pages/BingSearchPage.java  
│   │       │   ├── utils/WaitUtils.java  
│   │       │   ├── utils/ConfigReader.java  
│   │       └── listeners/TestListener.java  
│   └── test/  
│       ├── java/  
│       │   ├── com.example.tests/  
│       │   ├── SearchTest.java  
│       └── DataProviderClass.java  
└ resources/  
    ├── config.properties  
    └ log4j2.xml
```

## How to run

1. Import as a Maven project in your IDE.
  2. `mvn clean test -DthreadCount=3`
  3. Or run via TestNG using `testng.xml`.
-

## pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>selenium-framework</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <selenium.version>4.11.0</selenium.version>
    <testng.version>7.8.0</testng.version>
  </properties>

  <dependencies>
    <!-- Selenium -->
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>${selenium.version}</version>
    </dependency>

    <!-- WebDriverManager -->
    <dependency>
      <groupId>io.github.bonigarcia</groupId>
      <artifactId>webdrivermanager</artifactId>
      <version>5.4.0</version>
    </dependency>

    <!-- TestNG -->
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>${testng.version}</version>
      <scope>test</scope>
    </dependency>

    <!-- ExtentReports (optional) -->
    <dependency>
      <groupId>com.aventstack</groupId>
      <artifactId>extentreports</artifactId>
```

```

        <version>5.0.9</version>
    </dependency>

    <!-- Apache POI (optional for Excel) -->
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi-ooxml</artifactId>
        <version>5.2.3</version>
    </dependency>

    <!-- Jackson for JSON (optional) -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.15.2</version>
    </dependency>

    <!-- Logging -->
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.20.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.20.0</version>
    </dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.1.2</version>
            <configuration>
                <suiteXmlFiles>
                    <suiteXmlFile>testng.xml</suiteXmlFile>
                </suiteXmlFiles>
                <parallel>methods</parallel>
                <threadCount>${threadCount}</threadCount>
            </configuration>
        </plugin>
    </plugins>
</build>

```

```
</project>
```

## config.properties (resources)

```
baseUrl=https://www.bing.com  
defaultTimeout=10
```

## BrowserFactory.java

```
package com.example.framework.base;  
  
import io.github.bonigarcia.wdm.WebDriverManager;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
import org.openqa.selenium.chrome.ChromeOptions;  
  
public class BrowserFactory {  
  
    public static WebDriver createChromeWithProfile() {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions options = new ChromeOptions();  
  
        // Reduce detection  
        options.setExperimentalOption("excludeSwitches", new String[]{"enable-  
automation"});  
        options.setExperimentalOption("useAutomationExtension", false);  
        options.addArguments("--disable-blink-features=AutomationControlled");  
  
        // Use user profile if you want (optional)  
        // options.addArguments("user-data-dir=C:/Users/yourUser/AppData/Local/  
Google/Chrome/User Data");  
  
        return new ChromeDriver(options);  
    }  
}
```

## BaseTest.java

```
package com.example.framework.base;

import com.example.framework.utils.ConfigReader;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;

import java.time.Duration;

public class BaseTest {
    protected WebDriver driver;
    protected WebDriverWait wait;
    protected ConfigReader config = new ConfigReader();

    @BeforeMethod(alwaysRun = true)
    public void setUp() {
        driver = BrowserFactory.createChromeWithProfile();
        driver.manage().window().maximize();
        wait = new WebDriverWait(driver,
Duration.ofSeconds(config.getInt("defaultTimeout")));
    }

    @AfterMethod(alwaysRun = true)
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

---

## BingSearchPage.java (Page Object)

```
package com.example.framework.pages;

import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
```

```

import java.time.Duration;
import java.util.List;
import java.util.stream.Collectors;

public class BingSearchPage {
    private WebDriver driver;
    private WebDriverWait wait;

    private By searchBox = By.name("q");
    private By resultTitles = By.xpath("//h2"); // Bing uses h2 for titles in
results

    public BingSearchPage(WebDriver driver, WebDriverWait wait) {
        this.driver = driver;
        this.wait = wait;
    }

    public void open() {
        driver.get("https://www.bing.com");
    }

    public void search(String query) {
        WebElement box =
wait.until(ExpectedConditions.visibilityOfElementLocated(searchBox));
        box.clear();
        box.sendKeys(query + Keys.ENTER);
        wait.until(ExpectedConditions.titleContains(query));
    }

    public List<String> getResultTitles() {
        wait.until(ExpectedConditions.visibilityOfElementLocated(resultTitles));
        List<String> titles = driver.findElements(resultTitles)
            .stream()
            .map(WebElement::getText)
            .filter(s -> !s.isEmpty())
            .collect(Collectors.toList());
        return titles;
    }
}

```

## WaitUtils.java

```
package com.example.framework.utils;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;

public class WaitUtils {

    public static WebElement waitForVisibility(WebDriver driver, By locator,
int seconds) {
        WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(seconds));
        return
wait.until(ExpectedConditions.visibilityOfElementLocated(locator));
    }
}
```

## ConfigReader.java

```
package com.example.framework.utils;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class ConfigReader {
    private Properties props = new Properties();

    public ConfigReader() {
        try (InputStream is =
getClass().getClassLoader().getResourceAsStream("config.properties")) {
            if (is != null) props.load(is);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

    public String get(String key) {
        return props.getProperty(key);
    }

    public int getInt(String key) {
        return Integer.parseInt(props.getProperty(key));
    }
}

```

## DataProviderClass.java

```

package com.example.tests;

import org.testng.annotations.DataProvider;

public class DataProviderClass {

    @DataProvider(name = "searchData", parallel = true)
    public static Object[][] getData() {
        return new Object[][]{
            {"Selenium WebDriver"},
            {"TestNG Parallel

### 6.1 Cross-Browser Support (Chrome, Edge, Firefox)

(Add same content as previous...)
            Execution"},
            {"Java Automation"}
        };
    }
}

```

## SearchTest.java

```

package com.example.tests;

import com.example.framework.base.BaseTest;
import com.example.framework.pages.BingSearchPage;
import org.testng.annotations.Test;

import java.util.List;

```



```

public class SearchTest extends BaseTest {

    @Test(dataProvider = "searchData", dataProviderClass =
DataProviderClass.class)
    public void searchAndPrintResults(String query) {
        BingSearchPage page = new BingSearchPage(driver, wait);
        page.open();
        page.search(query);

        List<String> titles = page.getResultTitles();
        System.out.println("Results for: " + query);
        int limit = Math.min(5, titles.size());
        for (int i = 0; i < limit; i++) {
            System.out.println((i + 1) + ") " + titles.get(i));
        }
    }
}

```

## TestListener.java (captures screenshot on failure)

```

package com.example.framework.listeners;

import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

import java.io.File;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class TestListener implements ITestListener {

    @Override
    public void onTestFailure(ITestResult result) {
        Object obj = result.getInstance();
        try {
            WebDriver driver = (WebDriver)
result.getTestContext().getAttribute("driver");

```

```

        if (driver == null) return;
        File src = ((TakesScreenshot)
driver).getScreenshotAs(OutputType.FILE);
        String ts =
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMdd_HHmmss"));
        File dest = new File("screenshots/" + result.getName() + "_" + ts +
".png");
        FileUtils.copyFile(src, dest);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@Override public void onStart(ITestContext context) {}
@Override public void onFinish(ITestContext context) {}
@Override public void onTestStart(ITestResult result) {}
@Override public void onTestSuccess(ITestResult result) {}
@Override public void onTestSkipped(ITestResult result) {}
@Override public void onTestFailedButWithinSuccessPercentage(ITestResult
result) {}
}

```

## testng.xml

```

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Parallel Suite" parallel="tests" thread-count="3">
    <listeners>
        <listener class-name="com.example.framework.listeners.TestListener"/>
    </listeners>

    <test name="Bing Search Tests">
        <classes>
            <class name="com.example.tests.SearchTest"/>
        </classes>
    </test>
</suite>

```

## Jenkinsfile (simple)

```

pipeline {
    agent any

```

```
stages {
  stage('Checkout') {
    steps { checkout scm }
  }
  stage('Build & Test') {
    steps {
      sh 'mvn clean test -DthreadCount=3'
    }
  }
}
post {
  always { archiveArtifacts artifacts: 'screenshots/**, target/surefire-
reports/**', allowEmptyArchive: true }
}
```

---

## Notes & tips

- Use **Bing** to avoid Google's anti-bot checks.
- For a more stealthy Chrome use a dedicated real profile (see BrowserFactory).
- Add ExtentReports in the listener for HTML reports.
- If running in CI, ensure Chrome/driver are available or use a Docker image with GUI (selenium/standalone-chrome) or headless options.

---

If you want, I can: - generate a downloadable ZIP of this framework, - convert it to a GitHub repo with commit history, - or add ExtentReports integration + sample report.

Tell me which next step you want.