

TestNG Reporting

In this lesson, you'll understand TestNG Reporting.

WE'LL COVER THE FOLLOWING ^

- List of default TestNG reports
- Custom reporters

List of default TestNG reports

TestNG adds a few basic reports of the test run by default. The following are some reporters:

- `org.testng.reporters.SuiteHTMLReporter` – generates a HTML reporter for suites.
- `org.testng.reporters.FailedReporter` – generates *testng-failed.xml* containing only the failed tests.
- `org.testng.reporters.XMLReporter` – generates summary of test output in xml format.
- `org.testng.reporters.EmailableReporter2` – generates a single-page HTML report *emailable-report.html* of the test results.
- `org.testng.reporters.JUnitReportReporter` – generates Junit test output xml for each of the test suites.

Custom reporters

We can have our own reporters by implementing `org.testng.IReporter`.

Sample Implementation of `IReporter` for generating a summary report of the test run:

```
package com.example;
```

```
import java.io.File;
```

```

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.text.DecimalFormat;
import java.util.List;

import org.testng.IReporter;
import org.testng.ISuite;
import org.testng.ITestContext;
import org.testng.xml.XmlSuite;

public class SummaryReport implements IReporter {

    private DecimalFormat decimalFormat = new DecimalFormat("00.##");

    @Override
    public void generateReport(List<XmlSuite> xmlSuites, List<ISuite> suites, String outputDirectory) {

        File file = new File(outputDirectory, "summary.html");
        try (Writer writer = new FileWriter(file)) {
            writer.write("<html><body>");
            String title = "<table border='1' style='background-color:yellow;border:1px dotted black;width:80%;border-collapse:collapse;'><caption style='margin: 1em; font-weight: bolder'>Test Execution Summary Report</caption><tbody>";
            writer.write(title);
            writer.write(
                "<tr><th style='padding:3px;'align='center'>Suite Name</th><th style='padding:3px;'align='center'>Total</th><th style='padding:3px;'align='center'>Passed</th><th style='padding:3px;'align='center'>Failed</th><th style='padding:3px;'align='center'>Skipped</th><th style='padding:3px;'align='center'>% Tests Passed</th><th style='padding:3px;'align='center'>Total Execution Time (Sec)</th></tr>");

            for (ISuite suite : suites) {
                suite.getResults().forEach((k, v) -> {

                    ITestContext context = v.getTestContext();
                    Integer passed = context.getPassedTests().size();
                    Integer failed = context.getFailedTests().size();
                    Integer skipped = context.getSkippedTests().size();
                    Integer total = passed + failed + skipped;
                    Double execTime = (context.getEndDate().getTime() - context.getStartDate().getTime()) * 0.001;

```

```

        if (total > 0) {
            try {

                String percent = decimalFormat
                    .format((passed.doubleValue() / total.
doubleValue()) * 100.0D);

                writer.write(String.format(
                    "<tr><td style='text-align: center'>%s
</td><td style='text-align: center'>%s</td><td style='text-align: cente
r'>%s</td><td style='text-align: center'>%s</td><td style='text-align: cen
ter'>%s</td><td style='text-align: center'>%s</td><td style='text-align: c
enter'>%s</td></tr>",

                    suite.getName(), total, passed, failed
, skipped, percent,

                    decimalFormat.format(execTime)));
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
}

writer.write("</tbody></table></body></html>");
} catch (IOException ex) {
    ex.printStackTrace();
}

System.out.println("writing summary report to " + file.getAbsolute
Path());
}
}

```

This custom implementation of `IReporter` needs to be added to *testng.xml* like:

```

<listeners>
    <listener class-name="com.example.SummaryReportListener" />
</listeners>

```

Or in test class like:

```

@Listeners({ com.example.SummaryReportListener.class })
public class TestClass {}

```

Adding listeners is already discussed in the [TestNG Listener](#) lesson.

The default test output folder is `${user.dir}/test-output`.

Custom Reporter produces HTML report *summary.html* like below:

Test Execution Summary Report

Suite Name	Total	Passed	Failed	Skipped	% Tests Passed	Total Execution Time (Sec)
Default suite	2	1	1	0	50	00.18

Below is the sample of *emailable-report.html*:

Test	# Passed	# Skipped	# Retried	# Failed	Time (ms)	Included Groups	Excluded Groups
Surefire suite							
Surefire test	1	0	0	0	40		

Class	Method	Start	Time (ms)
Surefire suite			
Surefire test — passed			
com.mycompany.app.AppTest	shouldAnswerWithTrue	1577544119390	5

Surefire test

com.mycompany.app.AppTest#shouldAnswerWithTrue

[back to summary](#)

In this lesson, we learned about reporters in TestNG. In the next lesson, you'll learn about integrating the Allure reporting framework.