Automatic Logging using Aspects

In this lesson, we will see how to use AspectJ for logging all assertions done using TestNG Assert and SoftAssert.

WE'LL COVER THE FOLLOWING

What is AspectJ?

AspectJ is a seamless aspect-oriented extension to the Java programming language that enables clean modularization of crosscutting concerns, such as error checking and handling, synchronization, context-sensitive behavior, performance optimizations, monitoring and logging, debugging support, and multi-object protocols.

How can we use AspectJ for automatic logging?

We can use `AspectJ` for automatically logging whenever we are performing assertions using `org.testng.Assert` (stops further execution of the test on failure) or `org.testng.asserts.SoftAssert` (continues test execution, irrespective of whether the assertions are passing or failing. At the completion of the test execution, TestNG will show all the occurred failures, if any).

This strategy can be applied to various other operations as well.

To know more about `AspectJ`, please follow the link.

Dependency

Apart from adding the dependency, we need to attach the dependency to javaagent also.

Maven

```
<properties>
   <aspectj.version>1.9.5</aspectj.version>
</properties>


<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>${aspectj.version}</version>
</dependency>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.2</version>
            <configuration>
                <testFailureIgnore>false</testFailureIgnore>
                <argLine>
                    -javaagent:"${settings.localRepository}/org/aspectj/as
pectjweaver/${aspectj.version}/aspectjweaver-${aspectj.version}.jar"
                </argLine>
            </configuration>
            <dependencies>
                <dependency>
                    <groupId>org.aspectj</groupId>
                    <artifactId>aspectjweaver</artifactId>
                    <version>${aspectj.version}</version>
                </dependency>
            </dependencies>
        </plugin>
    </plugins>
</build>
```

Gradle

```
configurations {
```

```
    agent
}

dependencies {
    agent group: 'org.aspectj', name: 'aspectjweaver', version: '1.9.5'

}

test.doFirst {
    jvmArgs "-javaagent:${configurations.agent.singleFile}"
}
```

Creating `@Aspect`

If we don't use `AspectJ`, we may have to explicitly log all the assertions and their arguments for debugging.

The below `LoggerAspect` class shows how to automatically log all the assertions along with their arguments using `AspectJ`.

You can think of `AspectJ` as an interceptor to help log certain methods (as given in the `@Pointcut` expression) automatically.

```
package com.example;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.reflect.MethodSignature;

@Aspect
public class LoggerAspect {

    private static final Logger LOG = LoggerFactory.getLogger(LoggerAspect
.class);

    @Pointcut("execution(* org.testng.Assert.assert*(..))")
    public void assertMethod() {
    }

    @Pointcut("execution(* org.testng.asserts.SoftAssert.assert*(..))")
    public void softAssertMethod() {
    }
```

```
    @Around("assertMethod() || softAssertMethod()")
    public Object logAsserts(final ProceedingJoinPoint joinPoint) throws T
hrowable {
        final MethodSignature methodSignature = (MethodSignature) joinPoin
t.getSignature();
        LOG.debug("assertions ::: calling {} with arguments {}", methodSig
nature.getMethod().getName(),
                java.util.Arrays.deepToString(joinPoint.getArgs()));
        return joinPoint.proceed();
    }
}
```

Creating `aop-ajc.xml`

`aop-ajc.xml` is the configuration file that contains all the `@Aspect` classes to be used by the `AspectJ` library. Make sure `aop-ajc.xml` is present under `src/main/resources/META-INF` path.

```
<aspectj>
    <aspects>
        <aspect name="com.example.LoggerAspect" />
    </aspects>
</aspectj>
```

Now you are familiar with logging to console and file using Logback and automatically logging using AspectJ. In the next section, you'll learn about reporting.