Using TestNG

Unsupported TestNG versions

- TestNG 5.14.3: Bad formatted pom.xml. - TestNG 5.14.4 and 5.14.5: TestNG is using a missing dependency (org.testng:guice:2.0). Excluding it, may break some features.

Configuring TestNG

To get started with TestNG, include the following dependency in your project (replacing the version with the one you wish to use):

```
<dependencies>
  [...]
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>6.9.8</version>
      <scope>test</scope>
    </dependency>
  [...]
</dependencies>
```

If you are using an older version of TestNG (<= 5.11), the dependency would instead look like this:

```
<dependencies>
  [...]
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>5.11</version>
      <scope>test</scope>
      <classifier>jdk15</classifier>
    </dependency>
  [...]
</dependencies>
```

Note: if you are using JDK 1.4 Javadoc annotations for your TestNG tests, replace the classifier `jdk15` with `jdk14` above.

This is the only step that is required to get started - you can now create tests in your test source directory (e.g., `src/test/java`). As long as they are named in accordance with the defaults such as `*Test.java` they will be run by Surefire as TestNG tests.

If you'd like to use a different naming scheme, you can change the `includes` parameter, as discussed in the Inclusions and Exclusions of Tests (./inclusion-exclusion.html) example.

Using Suite XML Files

Another alternative is to use TestNG suite XML files. This allows flexible configuration of the tests to be run. These files are created in the normal way, and then added to the Surefire Plugin configuration:

```
<plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          <suiteXmlFiles>
            <suiteXmlFile>testng.xml</suiteXmlFile>
          </suiteXmlFiles>
        </configuration>
      </plugin>
    [...]
</plugins>
```

This configuration will override the includes and excludes patterns and run all tests in the suite files.

Specifying Test Parameters

Your TestNG test can accept parameters with the `@Parameters` annotation. You can also pass parameters from Maven into your TestNG test, by specifying them as system properties, like this:

```
<plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          <systemPropertyVariables>
            <propertyName>firefox</propertyName>
          </systemPropertyVariables>
        </configuration>
      </plugin>
    [...]
</plugins>
```

For more information about setting system properties in Surefire tests, see System Properties (./system-properties.html).

Using Groups

TestNG allows you to group your tests. You can then execute one or more specific groups. To do this with Surefire, use the `groups` parameter, for example:

```
<plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          <groups>functest,perftest</groups>
        </configuration>
      </plugin>
    [...]
</plugins>
```

Likewise, the `excludedGroups` parameter can be used to run all but a certain set of groups.

Running Tests in Parallel

TestNG allows you to run your tests in parallel, including JUnit tests. To do this, you must set the `parallel` parameter, and may change the `threadCount` parameter if the default of 5 is not sufficient. For example:

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          <parallel>methods</parallel>
          <threadCount>10</threadCount>
        </configuration>
      </plugin>
    [...]
</plugins>
```

This is particularly useful for slow tests that can have high concurrency, or to quickly and roughly assess the independence and thread safety of your tests and code.

TestNG 5.10 and plugin version 2.19 or higher allows you to run methods in parallel test using data provider.

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          <properties>
            <property>
              <name>parallel</name>
              <value>methods</value>
            </property>
            <property>
              <name>dataproviderthreadcount</name>
              <value>30</value>
            </property>
          </properties>
        </configuration>
      </plugin>
    [...]
</plugins>
```

TestNG 6.9.8 (JRE 1.7) and plugin version 2.19 or higher allows you to run suites in parallel.

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          <suiteXmlFiles>
            <file>src/test/resources/testng1.xml</file>
            <file>src/test/resources/testng2.xml</file>
          </suiteXmlFiles>
          <properties>
            <property>
              <name>suitethreadpoolsize</name>
              <value>2</value>
            </property>
          </properties>
        </configuration>
      </plugin>
    [...]
</plugins>
```

See also Fork Options and Parallel Test Execution (./fork-options-and-parallel-execution.html).

Using Custom Listeners and Reporters

TestNG provides support for attaching custom listeners, reporters, annotation transformers and method interceptors to your tests. By default, TestNG attaches a few basic listeners to generate HTML and XML reports.

Unsupported versions: - TestNG 5.14.1 and 5.14.2: Due to an internal TestNG issue, listeners and reporters are not working with TestNG. Please upgrade TestNG to version 5.14.9 or higher. Note: It may be fixed in a future surefire version.

You can configure multiple custom listeners like this:

```
<dependencies>
[...]
  <dependency>
    <groupId>your-testng-listener-artifact-groupid</groupId>
    <artifactId>your-testng-listener-artifact-artifactid</artifactId>
    <version>your-testng-listener-artifact-version</version>
    <scope>test</scope>
  </dependency>
[...]
</dependencies>
[...]
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          <properties>
            <property>
              <name>usedefaultlisteners</name>
              <value>false</value> <!-- disabling default listeners is optional -->
            </property>
            <property>
              <name>listener</name>
              <value>com.mycompany.MyResultListener,com.mycompany.MyAnnotationTransformer,com.myc
ompany.MyMethodInterceptor</value>
            </property>
            <property>
              <name>reporter</name>
              <value>listenReport.Reporter</value>
            </property>
          </properties>
        </configuration>
      </plugin>
    [...]
</plugins>
```

For more information on TestNG, see the TestNG web site (http://www.testng.org) .

You can implement TestNG listener interface `org.testng.ITestListener` in a separate test artifact `your-testng-listener-artifact` with scope=test, or in project test source code `src/test/java`. You can filter test artifacts by the parameter `dependenciesToScan` to load its classes in current ClassLoader of surefire-testng provider.

The TestNG reporter class should implement `org.testng.IReporter`.

The level of verbosity

Since the plugin version 2.19 or higher the verbosity level can be configured in provider property `surefire.testng.verbose`. The verbosity level is between 0 and 10 where 10 is the most detailed. You can specify -1 and this will put TestNG in debug mode (no longer slicing off stack traces and all). The default level is 0.

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          [...]
          <properties>
            <property>
              <name>surefire.testng.verbose</name>
              <value>10</value>
            </property>
          </properties>
          [...]
        </configuration>
      </plugin>
    [...]
</plugins>
```

Customizing TestNG Object Factory

Since the plugin version 2.19 and TestNG 5.7 or higher you can customize TestNG object factory by implementing `org.testng.IObjectFactory` and binding the class name to the key `objectfactory` in provider properties:

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          [...]
          <properties>
            <property>
              <name>objectfactory</name>
              <value>testng.objectfactory.TestNGCustomObjectFactory</value>
            </property>
          </properties>
          [...]
        </configuration>
      </plugin>
    [...]
</plugins>
```

Customizing TestNG Test Runner Factory

Since the plugin version 2.19 and TestNG 5.9 or higher you can customize TestNG runner factory by implementing `org.testng.ITestRunnerFactory` and binding the class name to the key `testrunfactory` in provider properties:

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          [...]
          <properties>
            <property>
              <name>testrunfactory</name>
              <value>testng.testrunnerfactory.TestNGCustomTestRunnerFactory</value>
            </property>
          </properties>
          [...]
        </configuration>
      </plugin>
    [...]
</plugins>
```

Running 'testnames' in test tag

Only tests defined in a `test` tag matching one of these names will be run. In this example two tests run out of 7; namely *InstallTest* and *ATest*. The test *a-t3* does not match any test in *suite.xml*.

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          [...]
          <suiteXmlFiles>
            <file>src/test/resources/suite.xml</file>
          </suiteXmlFiles>
          <properties>
            <property>
              <name>testnames</name>
              <value>a-t1,a-t3</value>
            </property>
          </properties>
          [...]
        </configuration>
      </plugin>
    [...]
</plugins>
```

The test suite 'suite.xml' :

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Component Tests" verbose="2" annotations="JDK">
  <test name="a-t1" preserve-order="true" >
    <classes>
      <class name="server.InstallTest" />
      <class name="server.ATest" />
    </classes>
  </test>
  <test name="a-t2" preserve-order="true" >
    <classes>
      <class name="server.SCHTest" />
      <class name="server.PRGTest" />
      <class name="server.SIBBTest" />
      <class name="server.EDNTest" />
      <class name="server.PPVTest" />
    </classes>
  </test>
</suite>
```

Running TestNG and JUnit Tests

TestNG 6.5.1 and higher provides support to run TestNG and JUnit 4.x in current Maven project. All you need is to introduce TestNG and JUnit dependency in POM.

```
<dependencies>
  [...]
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>6.9.4</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.10</version>
    <scope>test</scope>
  </dependency>
  [...]
</dependencies>
```

You may want to run two providers, e.g. `surefire-junit47` and `surefire-testng`, and avoid running JUnit tests within `surefire-testng` provider by setting property `junit=false`.

```
<plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
          [...]
          <properties>
            <property>
              <name>junit</name>
              <value>false</value>
            </property>
          </properties>
          <threadCount>1</threadCount>
          [...]
        </configuration>
        <dependencies>
          <dependency>
            <groupId>org.apache.maven.surefire</groupId>
            <artifactId>surefire-junit47</artifactId>
            <version>3.0.0-M5</version>
          </dependency>
          <dependency>
            <groupId>org.apache.maven.surefire</groupId>
            <artifactId>surefire-testng</artifactId>
            <version>3.0.0-M5</version>
          </dependency>
        </dependencies>
      </plugin>
    [...]
</plugins>
```

Privacy Policy (https://maven.apache.org/surefire/maven-surefire-plugin/privacy-policy.html)