Using TestNG

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>6.8.8</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 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>2.18.1</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>2.18.1</version>
        <configuration>
          <systemPropertyVariables>
            <propertyName>firefox</propertyName>
          </systemPropertyVariables>
        </configuration>
      </plugin>
    [...]
</plugins>
```

For more information about setting system properties in Surefire tests, see System Properties.

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>2.18.1</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>2.18.1</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.

See also Fork Options and Parallel Test Execution.

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.

You can configure multiple custom listeners like this:

```
</plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.18.1</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.my
company.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.