About time-test

time-test is a lightweight framework for testing time-based functionality.

GitHub Repository	https://github.com/Devexperts/time-test
Public Maven repo	https://bintray.com/devexperts/Maven/time-test
License	GPLv3
Contact	dxlab@devexperts.com

README.md (source from time-test)

time-test

time-test is a framework for testing time-based functionality.

Sometimes you have code with similar logic: "wait for 30 seconds and then do something if no changes have been detected".

If you want to test such logic you can write something like this:

```
Thread.sleep(30_000 + 200);
check();
```

However, such tests are undesirable to be in the project. Also what if you want to test logic which should be executed once per month? Or what if this "something doing" hasn't done before the check starts?

time-test provides an easy way to test such logic. For example, the code above can be written more clear:

```
TestTimeProvider.increaseTime(30_000);
TestTimeProvider.waitUntilThreadsAreFrozen(200 /*ms*/);
check();
```

And this test doesn't do superfluous work.

time-test instruments byte-code and change time-based methods invocations (such as System.currentTimeMillis, Object.wait, Unsafe.park) to our own implementation.

TimeProvider

This interface provides an implementation of time-based methods. The default implementation forwards all calls to standard system methods.

Use `XXXTimeProvider.start()` to start using it. Don't forget to reset TimeProvider to default. Use `XXXTimeProvider.reset()` for this.

DummyTimeProvider

DummyTimeProvider throws `UnsupportedOperationException` on all method calls. It should be used for testing functionality which does not depend on time.

TestTimeProvider

TestTimeProvider provides full access on time. Use `TestTimeProvider.setTime(millis)` and `TestTimeProvider.inscreaseTime(millis)` to change current time. Use `TestTimeProvider.waitUntilThreadsAreFrozen` to wait until all threads complete their work.

In order to work properly **TestTimeProvider** defines if it is executed in the testing code or not on every time-based operation invocation (including `Object.notity()` and similars). For this purpose an entry point to your code have to be specified (see **timetest.testingCode** property). After that, if your code starts a thread it will be marked as ours too (**time-test** traces Thread.start() invocations for this purpose). However, there are some problems if you use shared scheduler like `ForkJoinPool`. In order to work with it expand **timetest.testingCode** property.

Configuration

You can pass your own configuration in `timetest.properties` properties file or set these properties as system parameters (*-Dparam.name=value*). The `timetest.properties` file should be in the application classpath.

- **timetest.testingCode** - defines the entry points of the testing code in glob format. Several globs can be separated by comma. Default value: *com.devexperts.*Test* (all classes with *Test* suffix).
- **timetest.nonTestingCode** - defines the scope of code which have to be processed like non-testing onecode in glob format. It can be helpful to print real timestamps in logging instead of virtual ones. Default value: *com.devexperts.logging.**.
- ***timetest.log.level*** defines internal logging level. Possible values: *DEBUG*, *INFO* (default value), *WARN*, *ERROR*.
- ***timetest.log.file*** defines path of file to be used for logging. By default logs are printed to the standard output.
- ***timetest.cache.dir*** [experimental] defines directory to be used for transformed classes caching. This feature is unstable, use it on your own risk.
- ***timetest.include*** defines the transformation scope using globs. For example, setting the value to `package.to.transform.*,another.package.to.transform.*` informs **time-test** to transform classes from these packages only. By default all classes are included.
- ***timetest.exclude*** defines the classes which should be excluded from transformation. The syntax is similar to **timetest.include** option. Default value: *org.apache.maven.*,org.junit.*,com.devexperts.test.**

Maven

Time-test is implemented as java agent and you should copy it to build directory and configure *maven-surefire-plugin* to use it for tests.

```
...
<plugins>
    <!-- maven-dependency-plugin is used to copy "timetest" agent into target directory -->
    <plugin>
        <artifactId>maven-dependency-plugin</artifactId>
        <executions>
            <execution>
                <id>copy-timetest-agent</id>
                <phase>process-test-classes</phase>
                <goals>
                    <goal>copy</goal>
                </goals>
                <configuration>
                    <artifactItems>
                        <artifactItem>
                            <groupId>com.devexperts.timetest</groupId>
                            <artifactId>agent</artifactId>
                            <version>${project.version}</version>
                            <outputDirectory>${project.build.directory}</outputDirectory>
                            <destFileName>timetest.jar</destFileName>
                        </artifactItem>
                    </artifactItems>
                </configuration>
            </execution>
        </executions>
    </plugin>
    <!-- Configure maven-surefire-plugin to use "timetest" agent -->
    <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <configuration>
            <argLine>-javaagent:${project.build.directory}/timetest.jar</argLine>
        </configuration>
    </plugin>
</plugins>
...
```

Usage example

```
@Before
public void setUp() {
    TestTimeProvider.start();
}

@After
public void tearDown() {
    TestTimeProvider.reset();
}

@Test(timeout = 100)
public void testSleepWithTestTimeProvider() {
    Thread t = new Thread(() -> {
        // Do smth
        int sum = 0;
        for (int i = 0; i < 100_000; i++)
            sum += i;
        // Sleep
        Thread.sleep(10_000);
    });
    t.start();
    TestTimeProvider.waitUntilThreadsAreFrozen();
    TestTimeProvider.increaseTime(10_000);
    t.join();
}
```

Contacts

If you need help, you have a question, or you need further details on how to use **time-test**, you can refer to the following resources:

- dxLab research group at Devexperts
- GitHub issues

You can use the following e-mail to contact us directly: