Configuring the Jtest Plugin for Gradle

Initial Setup

Integration with Gradle is achieved with the following components:

- The Jtest Plugin for Gradle – shipped with Jtest in its local Maven repository in `[INSTALL_DIR]\integration\maven`
- The `jtest.gradle` and `init.gradle` scripts shipped with Jtest in `[INSTALL_DIR]\integration\gradle`

Configuring the Location of Jtest

The `jtest.gradle` and `init.gradle` scripts automatically detect the Jtest installation directory based on their own location. They set the detected directory as the `jtest.home` property, which is later used by the Jtest Plugin for Gradle to configure the `home` parameter.

If you don't use the scripts, or their location is different than the Jtest installation directory (for example, you moved the scripts or want to execute Jtest from another location), you need to configure the location of Jtest in one of the following ways:

- **The `jtest.home` property**. Provide the path to the Jtest installation directory in one of the following files:
  - `$HOME/.gradle/gradle.properties`
  - `PROJECT_DIR/gradle.properties`
- **JTEST_HOME environment variable**. Initscripts in **Gradle 2.0 and earlier** are unable to resolve Java system properties in initscript code blocks. Initscript `[INSTALL_DIR]/integration/gradle/init.gradle` will the Jtest location only if it is set as the JTEST_HOME environment variable.

See Types of Gradle Properties for details.

Configuring Jtest Execution

Before you start working with the Jtest Plugin for Gradle, configure the Jtest license and other settings in one of the following .properties files:

- `[INSTALL_DIR]/jtestcli.properties`
- `$HOME/jtestcli.properties`
- `$HOME/.gradle/gradle.properties`
- `[PROJECT_DIR]/gradle.properties`

See Setting the License and Configuration for details.

Configuring Analysis in the Command Line

The `init.gradle` script shipped with Jtest allows you to integrate with Gradle – without having to modify your Gradle build script.

Run Gradle with the `jtest` task, and provide the location to the script with the `-I` option to apply the plugin:

```
gradle jtest -I [INSTALL_DIR]/integration/gradle/init.gradle
```

If the `jtest.home` property is not configured, the script will automatically detect the Jtest installation directory based on their own location, and set the property accordingly.

Configuring Analysis in the Build Script

You can modify your build script to integrate with Gradle by applying the `jtest.gradle` script to the build script (Minimal Configuration) or configuring the Jtest task in the build script (Extended Configuration). Executing analysis with Jtest will depend on on how the build script is configured,

Minimal Configuration

The Jtest Plugin for Gradle can be applied in the Gradle build script by adding the path to the `jtest.gradle` script – directly in the root project build script:

```
//build script blocks
apply from: System.properties['jtest.home'] + '/integration/gradle/jtest.gradle'
//other build script blocks
```

If the `jtest.home` property is not configured, the script will automatically detect the Jtest installation directory based on their own location, and set the property accordingly.

Executing analysis

If you configure your project with minimal configuration, execute analysis with the following command:

```
gradle jtest -Djtest.home=[INSTALL_DIR]
```

Extended Configuration

1. Configure the jtest.home property in the `[PROJECT_DIR]/gradle.properties` or `$HOME/.gradle/gradle.properties` file . For example:

   ```
   # Java system property 'jtest.home
   systemProp.jtest.home=PATH/TO/JTEST
   # project property 'jtest.settings' (could be Java system property as well)
   jtest.settings=PATH/TO/SETTINGS/FILE
   ```

   As a result, you won't need to specify the path to the Jtest installation directory in your command line.
2. Configure the Jtest task in the `PROJECT_DIR/build.gradle` build script. For example:

   ```
   //build script blocks
   apply from: System.properties['jtest.home'] + '/integration/gradle/jtest.gradle'

   jtest {
     config = "builtin://Recommended Rules"
     showSettings = true
     resources = [
       "**/*.java",
       "my_project_name/src/**/*.xml"
     ]
     compilation {
       id = "main" //default value
       override = false //default value
       encoding = "MacRoman"
     }
   }
   // other build script blocks
   ```

Executing analysis

If you configure your project with minimal configuration, execute analysis with the following command:

```
gradle jtest
```

Types of Gradle Properties

Gradle supports two kinds of properties:

Java System Properties

The system properties can be configured:

- in the command line, with the `-D` parameter (`-DmyProperty`)
- in `gradle.properties` files, with `systemProp` (`systemProp.MY_PROPERTY=MY_VALUE`)

The system properties are accessible from the Gradle build script using `System.properties['myProperty']`.

Gradle Project Properties

The Gradle properties can be configured:

- in the command line, with the `-P` or -Dorg.gradle.project parameters (`-PmyProperty` or `-Dorg.gradle.project.myProperty`)
- in `gradle.properties` files

The Gradle properties are accessible by using their names directly in the Gradle build script, or by using the `property('myProperty')` function.

If your property key contains the "**.**" character, it should be accessed via `property('myProperty')` rather than directly in the build script.

See the Gradle User Manual for details (the System Properties and Gradle Properties sections).

Manual Customization of Compilation Data

If you need to modify compilation data for all projects, you can use the `-Djtest.dataUpdate` command line option.

See Compilation Data Model for more information about how to customize compilation data automatically detected by the Jtest plugins.