Gradle Training Exercises

01-Setup

1. Copy the Gradle distribution from the USB stick to your hard disk.
2. Download and unzip the Workshop Exercises from https://github.com/pniederw/gradle-workshop-gr8conf/zipball/master.
3. Add an environment variable `GRADLE_HOME` pointing to the top-level directory of the Gradle distribution.
4. Add `GRADLE_HOME/bin` to the `PATH` environment variable.
5. Activate the Gradle Daemon by setting the environment variable `GRADLE_OPTS` to `-Dorg.gradle.daemon=true`.
6. Open a terminal and execute `gradle -v`.
7. Have a look at `http://gradle.org/downloads.html`.

02-Quickstart

1. Go to `GRADLE_HOME/samples/java/quickstart`.
2. Check the output of `gradle help`, `gradle -?` and `gradle tasks`.
3. Build the archives for this Java project and try to find them.
4. Run the tests of this project with the Gradle UI. Start the UI with `gradle --gui`.

03-Incremental Build

1. Go to `GRADLE_HOME/samples/java/quickstart`.
2. Run `gradle clean build`.
3. Run `gradle build`. Observe that all tasks are up-to-date.
4. Add an arbitrary method to the project's main source code. Which tasks will be run again on the next build?
5. Add an arbitrary comment to the project's main source code. Which tasks will be run again on the next build?

04-Tasks

1. Add a `hello` task that prints 'hello world'.
2. Execute this task.
3. Add a `date` task that prints out the current date.
4. Execute this task.

05-Custom Tasks

1. Write a custom task class of type `Greeting`. Add a property `greeting` for the greeting text. Assign a default value to it. The task action should print the greeting text property. Add two tasks to your build script, both of type `Greeting`. One should assign a custom value to the greeting property. Execute both tasks.

06-Task Dependencies

1. Make the `date` task depend on the `hello` task.
2. Execute the `date` task.
3. Execute `gradle tasks --all`.
4. The `--dry-run` (or `-m`) command line option executes the build but disables all actions. Execute `date` with the `dry-run` option.
5. Add some top level `println` statements to the script.
6. Add a `println` statement to the configuration block of the `date` task.
7. Execute the `hello` task and analyze the output.

07-Applying Plugins

1. Apply the plugin `info.gradle` in the `plugins` directory to the build. Execute `gradle tasks` to see what task has been added. Execute the task.

08-Testing

1. Run the tests with `testReport` set to false. What do you see?
2. Run the tests with different settings for `forkEvery`. What do you see?
3. Run the tests with different settings for `maxParallelForks`. What do you see?
4. (Optional) Add a listener that, if a test fails, opens the test results XML file.

   Hint:

   1. The test results XML file can be found at `build/test-results/TEST-<test-class-name>.xml`.
   2. To execute an external command in Groovy, use `"command args".execute()`.
   3. Check the Javadoc of the `org.gradle.api.tasks.testing.Test.afterTest` method to learn about its arguments.

09-Dependencies

1. Add the Maven Central repository and a configuration named `mydeps`. Assign the `org.apache.httpcomponents:httpclient:4.0.3` dependency to `mydeps`.
2. Add a task `showDeps` that prints out the files of the `mydeps` configuration.
3. Add task `copyDeps` that copies the files of the `mydeps` configuration into the `build/deps` dir.
4. Execute `gradle dependencies`.

10-Multi-Project Builds

1. Investigate the structure of the multi project build. Execute the `build` task from the root project and observe what is happening.
2. Go to the `api` project. Execute `build` from there. Execute also `buildNeeded` and `buildDependent`. What is different compared to executing the build task?
3. Execute the `build` task of the `api` project from the root project directory.
4. Execute `gradle projects` from the root directory.
5. Execute `gradle projects` and `gradle :projects` from the services directory.
6. Execute `gradle tasks` and `gradle :api:tasks` from the root project directory.
7. Execute `gradle :services:webservice:properties` from the root project directory.
8. Execute `gradle --profile clean build` from the root project. Have a look at the profile report in `build/reports/profile`.