Java with Maven

Last updated 24 March, 2023 • 3 min read

TIP: Please make sure to read Getting started with programming tasks first.

You can start with our sample project which can be found on GitHub:

Open sample project.

Download sample project.

Technical details for Java with Maven support

Any **Maven** project might be used as a programming task. You can use any testing framework supported by Maven, this sample project uses JUnit for running unit tests.

Maven build will be executed with the following command:

```
mvn clean test
```

Note: Your code pack must contain a `pom.xml` file. Unit tests executed during the build might be used to perform an automatic evaluation of the candidate's answer

Automatic assessment

It is possible to automatically assess the solution posted by the candidate. Automatic assessment is based on Unit Tests results and Code Quality measurements.

All unit tests that are executed during the build will be detected by the **TalentScore** platform.

There are two kinds of unit tests:

1. **Candidate tests** - unit tests that are visible for the candidate during the test. Should be used to do only the basic verification and help the candidate to understand the requirements. Candidate tests WILL NOT be used to calculate the final score.
2. **Verification tests** - unit tests that are hidden from the candidate during the test. Files containing verification tests will be added to the project after the candidate finishes the test and will be executed during the verification phase. Verification tests result will be used to calculate the final score.

After the candidate finishes the test, our platform builds the project posted by the candidate and executes verification tests and static code analysis.

Devskiller project descriptor

The programming task can be configured with the Devskiller project descriptor file. Just create a `devskiller.json` file and place it in the root directory of your project. Here is an example project descriptor:

```
{
  "readOnlyFiles" : [ "pom.xml" ],
  "hiddenFiles" : [ "hidden_file.txt" ],
  "verification" : {
    "testNamePatterns" : [".*verify_pack.*"],
    "pathPatterns" : ["**src/test/**/verify_pack**"]
  }
}
```

You can find more details about `devskiller.json` descriptor in our documentation

Automatic verification with verification tests

To enable automatic verification of candidates' solutions, you need to define which tests should be treated as verification tests.

All files classified as verification tests will be removed from a project prepared for the candidate.

To define verification tests, you need to set two configuration properties in `devskiller.json` project descriptor:

- `testNamePatterns` - an array of RegEx patterns that should match all the test names of verification tests. Test name contains: `[package_name].[Class_name].[method_name]`. In our sample project, all verification tests are in `verify_pack` package, so the following pattern will be sufficient:

```
"testNamePatterns" : [".*verify_pack.*"]
```

- `pathPatterns` - an array of GLOB patterns that should match all the files containing verification tests. All the files that match defined patterns will be deleted from candidates' projects and will be added to the projects during the verification phase.

```
"pathPatterns" : ["**src/test/**/verify_pack**"]
```