Appendix 3: Setting up your Java Project

Creating a Project from Maven Template

Using Terminal Commands

1. In a terminal (linux or Mac) or command prompt (Windows), navigate to the folder you want to create the Java project. Type this command: `mvn archetype:generate`

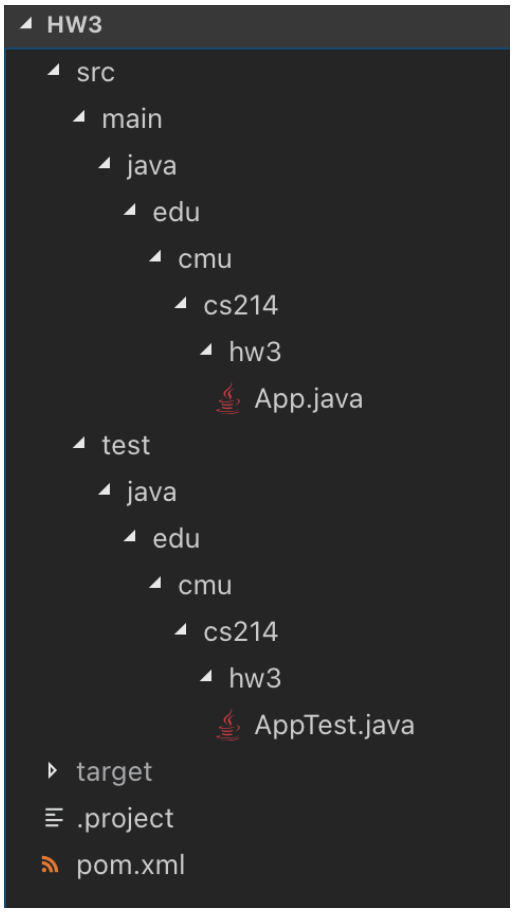
```
Eshitas-MacBook-Pro:17-214 eshitaagarwal$ mvn archetype:generate
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/maven-metadata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/maven-metadata.xml (980 B at 2.4 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/3.2.1/maven-archetype-plugin-3.2.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/3.2.1/maven-archetype-plugin-3.2.1.pom (12 kB at 184 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetype/maven-archetype/3.2.1/maven-archetype-3.2.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetype/maven-archetype/3.2.1/maven-archetype-3.2.1.pom (13 kB at 250 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/3.2.1/maven-archetype-plugin-3.2.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/3.2.1/maven-archetype-plugin-3.2.1.jar (101 kB at 1.2 MB/s)
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.2.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.2.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.2.1:generate (default-cli) @ standalone-pom ---
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetype/archetype-catalog/3.2.1/archetype-catalog-3.2.1.pom
```

2. Then in the terminal, specify a maven template. Recommended - `maven-archetype-quickstart`

3. Specify project packaging and project name.

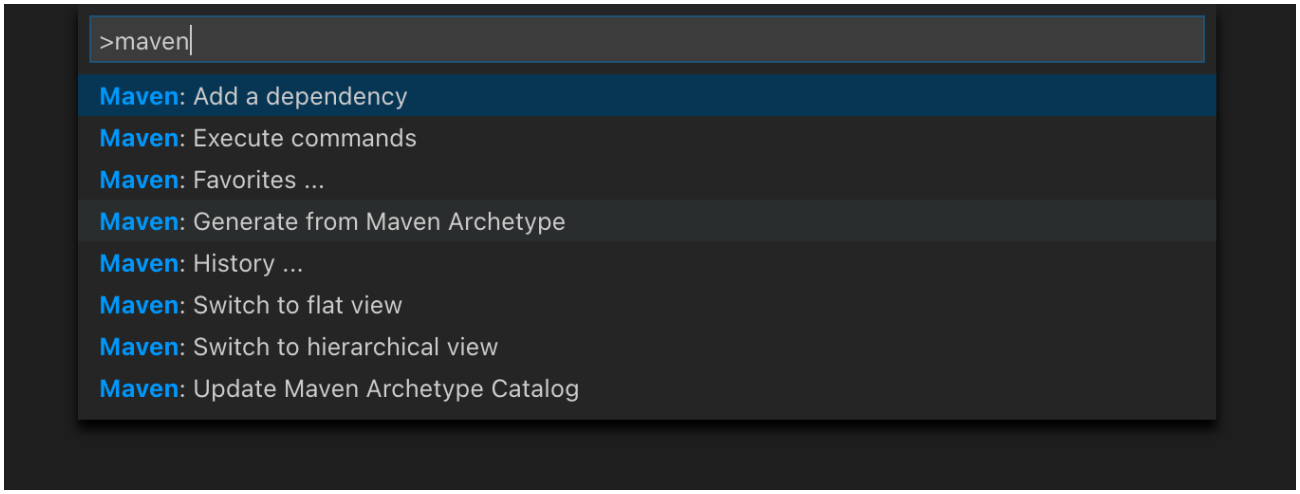
```
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1958: maven-archetype-quickstart
Choose archetype:
1: remote -> com.github.ywchang:maven-archetype-quickstart (Provide up-to-date java quickstart archetype)
2: remote -> com.haoxuer.maven.archetype:maven-archetype-quickstart (a simple maven archetype)
3: remote -> org.apache.maven.archetypes:maven-archetype-quickstart (An archetype which contains a sample Maven project.)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 3: 3
Choose org.apache.maven.archetypes:maven-archetype-quickstart version:
1: 1.0-alpha-1
2: 1.0-alpha-2
3: 1.0-alpha-3
4: 1.0-alpha-4
5: 1.0
6: 1.1
7: 1.3
8: 1.4
Choose a number: 8: 8
Define value for property 'groupId': edu.cmu.cs214
Define value for property 'artifactId': hw3
Define value for property 'version' 1.0-SNAPSHOT: : 1.0-SNAPSHOT
Define value for property 'package' edu.cmu.cs214: : edu.cmu.cs214.hw3
Confirm properties configuration:
groupId: edu.cmu.cs214
artifactId: hw3
version: 1.0-SNAPSHOT
package: edu.cmu.cs214.hw3
 Y: : Y
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.4
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: edu.cmu.cs214
[INFO] Parameter: artifactId, Value: hw3
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: edu.cmu.cs214.hw3
[INFO] Parameter: packageInPathFormat, Value: edu/cmu/cs214/hw3
[INFO] Parameter: package, Value: edu.cmu.cs214.hw3
[INFO] Parameter: groupId, Value: edu.cmu.cs214
[INFO] Parameter: artifactId, Value: hw3
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: /Users/eshitaagarwal/Desktop/17-214/hw3
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:04 min
[INFO] Finished at: 2022-09-20T13:08:25-04:00
[INFO] ------------------------------------------------------------------------
```

Above command will generate a Java project from `maven-archetype-quickstart` template. It looks like below:

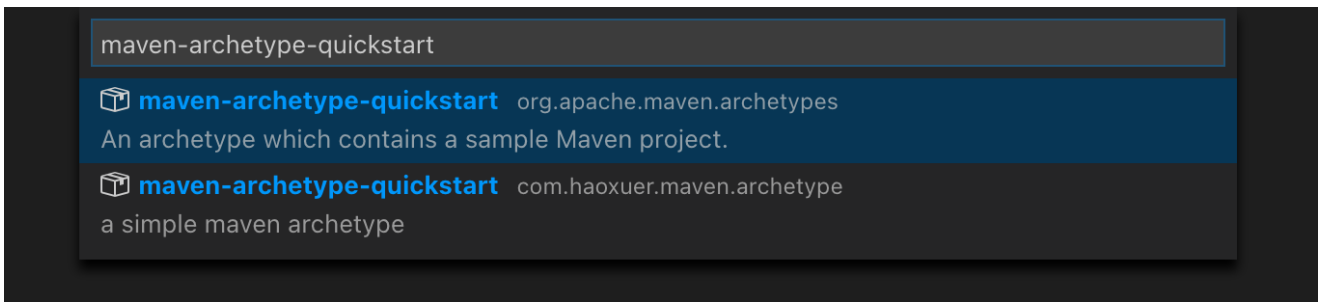


Using VS Code

1. Ensure that you have the Java Extension installed on your VS Code
2. Open the Command Palette and search for `maven`
3. Select `Maven: Generate from Maven Archetype`



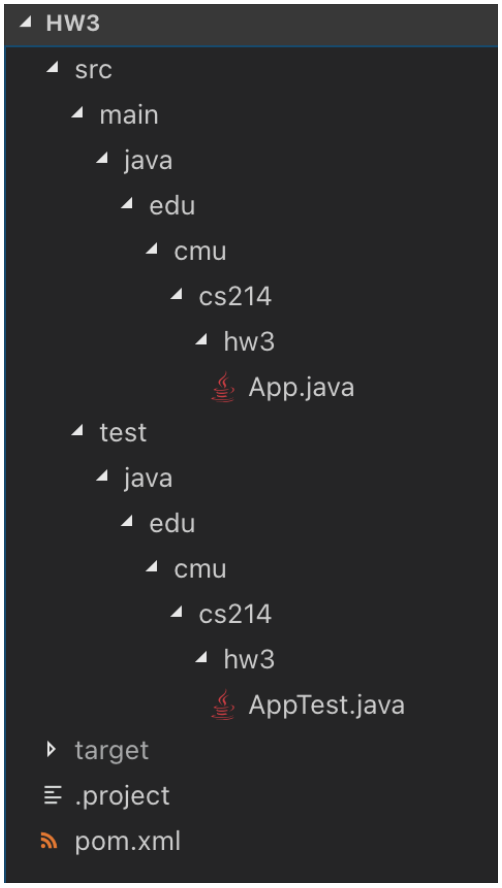
4. Specify a maven template. Recommended: `maven-archetype-quickstart`



5. Specify project packaging and project name

```
eshitas-mbp:17-214 eshitaagarwal$ mvn archetype:generate -DarchetypeArtifactId="maven-archetype-quickstart" -DarchetypeGroupId="org.apache.maven.archetypes"
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.2.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.2.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.2.1:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[INFO] Archetype [org.apache.maven.archetypes:maven-archetype-quickstart:1.4] found in catalog remote
Define value for property 'groupId': edu.cmu.cs214
Define value for property 'artifactId': hw3
Define value for property 'version' 1.0-SNAPSHOT: : 1.0-SNAPSHOT
Define value for property 'package' edu.cmu.cs214: : edu.cmu.cs214.hw3
Confirm properties configuration:
groupId: edu.cmu.cs214
artifactId: hw3
version: 1.0-SNAPSHOT
package: edu.cmu.cs214.hw3
 Y: : Y
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.4
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: edu.cmu.cs214
[INFO] Parameter: artifactId, Value: hw3
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: edu.cmu.cs214.hw3
[INFO] Parameter: packageInPathFormat, Value: edu/cmu/cs214/hw3
[INFO] Parameter: package, Value: edu.cmu.cs214.hw3
[INFO] Parameter: groupId, Value: edu.cmu.cs214
[INFO] Parameter: artifactId, Value: hw3
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: /Users/eshitaagarwal/Desktop/17-214/hw3
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  32.240 s
[INFO] Finished at: 2022-09-20T13:32:12-04:00
[INFO] ------------------------------------------------------------------------
```

Above steps will generate a Java project from `maven-archetype-quickstart` template. It looks like below:



Maven Command Cheat Sheet

- **validate:** validate the project is correct and all necessary information is available
- **compile:** compile the source code of the project
- **test:** test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- **package:** take the compiled code and package it in its distributable format, such as a JAR.
- **integration-test:** process and deploy the package if necessary into an environment where integration tests can be run
- **verify:** run any checks to verify the package is valid and meets quality criteria
- **install:** install the package into the local repository, for use as a dependency in other projects locally
- **deploy:** done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.
- **clean:** cleans up artifacts created by prior builds
- **site:** generates site documentation for this project

pom.xml

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project.

When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

Some of the configurations that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

The minimum requirement for a POM are the following:
`project` - root
`modelVersion` - should be set to 4.0.0
`groupId` - the id of the project's group
`artifactId` - the id of the project
`version` - the version of the artifact under the specified group
`<maven.compiler.source> and <maven.compiler.target>` - 17

Important - Use Java 17 for this assignment; Java 18 is not supported by GitHub actions

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.maven.example</groupId>
  <artifactId>hw3</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>hw3</name>
```

Based on the minimum POM, you can expand your configuration file by adding new elements, such as `<build>`, `<dependencies>` and `<reporting>`.
Check the documentation for their usages: [Maven – Introduction to the POM]

checkstyle.xml

The Checkstyle Plugin generates a report regarding the code style used by the developers. It needs a xml file containing predefined rulesets. Predefined rulesets available for use are:

sun_checks.xml and google_checks.xml. You can also copy the `checkstyle.xml` file from previous homework.

The plugin can be configured in the project's POM. An example configuration is below:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>3.7.1</version>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-project-info-reports-plugin</artifactId>
            <version>3.1.2</version>
        </plugin>
    </plugins>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-checkstyle-plugin</artifactId>
                <version>3.1.2</version>
                <configuration>
                    <failsOnError>true</failsOnError>
                    <consoleOutput>true</consoleOutput>
                </configuration>
                <executions>
                    <execution>
                        <id>validate</id>
                        <phase>validate</phase>
                        <goals>
                            <goal>check</goal>
                        </goals>
                    </execution>
                </executions>
                <dependencies>
                    <dependency>
                        <groupId>com.puppycrawl.tools</groupId>
                        <artifactId>checkstyle</artifactId>
```

```
                        <version>8.45</version>
                    </dependency>
                </dependencies>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
<reporting>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-checkstyle-plugin</artifactId>
            <version>3.1.2</version>
            <configuration>
                <configLocation>"path-to-your-checkstyle-file"/"checkstyle-
name".xml</configLocation>
            </configuration>
        </plugin>
        <!-- a plugin to include test results in the `mvn site` report -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-report-plugin</artifactId>
            <version>2.22.2</version>
        </plugin>
    </plugins>
</reporting>
```

.gitignore

Git sees every file in your working copy as one of three things:

- tracked - a file which has been previously staged or committed;
- untracked - a file which *has not* been staged or committed;
- ignored - a file which Git has been explicitly told to ignore.

Ignored files are usually build artifacts and machine generated files that can be derived from your repository source or should otherwise not be committed. Some common examples are:

- dependency caches, such as the contents of `/node_modules` or `/packages`
- compiled code, such as `.o` , `.pyc` , and `.class` files
- build output directories, such as `/bin` , `/out` , or `/target`

- files generated at runtime, such as `.log`, `.lock`, or `.tmp`
- hidden system files, such as `.DS_Store` or `Thumbs.db`
- personal IDE config files, such as `.idea/workspace.xml, .vscode`

In order to avoid committing unnecessary files, you should create a `.gitignore` file in the root directory of your repo. Example `.gitignore` file looks like below:

.gitignore

```
# Compiled class file
*.class

# Log file
*.log

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see
http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*

target/
.idea/

*.DS_Store
```

Quickstart for GitHub Actions

Creating Your workflow

1. Create a .github/workflows directory in your repository on GitHub if this directory does not already exist
2. In the .github/workflows directory, create a file named main.yml
3. Copy the following YAML contents into the main.yml file. This is an example from HW 1:

```yaml
# This is a basic workflow to help you get started with Actions

name: CI
```

```yaml
# Controls when the workflow will run
on:
 # Triggers the workflow on push or pull request events but only for the main branch
 push:

 # Allows you to run this workflow manually from the Actions tab
 workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in
parallel
jobs:
 # This workflow contains a single job called "build"
 build:
   # The type of runner that the job will run on
   runs-on: ubuntu-latest

   # Steps represent a sequence of tasks that will be executed as part of the job
   steps:
     # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
     - uses: actions/checkout@v2
       with:
         fetch-depth: 0

     # Checks-out Java
     - uses: actions/setup-java@v2
       with:
         distribution: microsoft
         java-version: 17

     # Checks-out Node
     - uses: actions/setup-node@v2
       with:
         node-version: 16

     # Identifies all relevant Java files that have been changed in latest commit
     - name: Change Java
       uses: tj-actions/changed-files@v13.1
       id: change-java
       with:
         files: |
           pom.xml
```

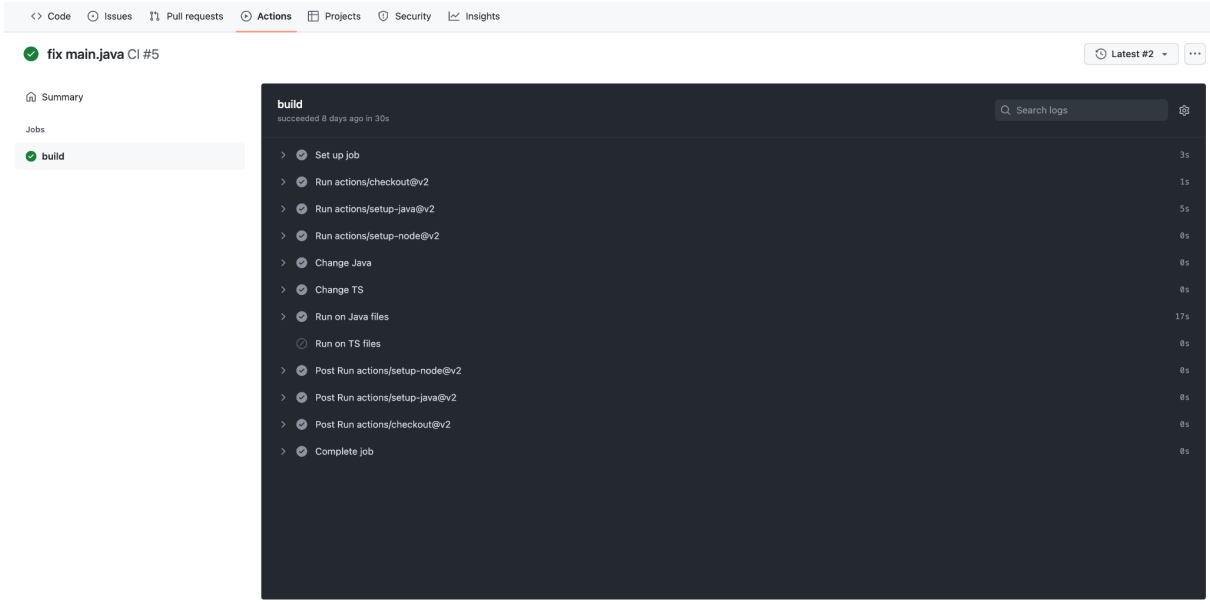
```
        **/*.java

    # Identifies all relevant TS files that have been changed in latest commit
    - name: Change TS
      uses: tj-actions/changed-files@v13.1
      id: change-ts
      with:
        files: |
          **/*.ts
          **/*.json

    # Runs type checker for Java files
    - name: Run on Java files
      if: steps.change-java.outputs.any_changed == 'true'
      run: |
        timeout 2m mvn -f java/pom.xml site

    # Runs compiler and linter for TS files
    - name: Run on TS files
      if: steps.change-ts.outputs.any_changed == 'true'
      run: |
        timeout 2m npm install typescript/
        timeout 2m npm run compile --prefix typescript/
        timeout 2m npm run lint --prefix typescript/
```

For your project, please follow the documentation to learn and customize your own actions. You can also start with reviewing and modifying the `.github/workflows/main.yml` file in your previous homework repo.

4. Commit and push this file

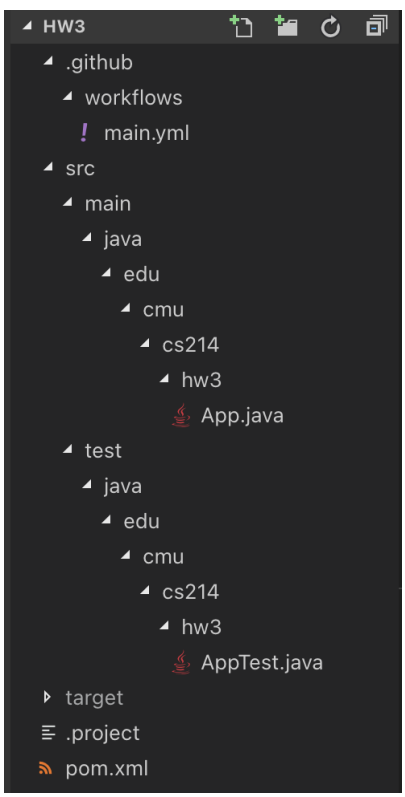
Viewing Your Workflow Results

- On GitHub.com, navigate to the main page of the repository.
- Under your repository name, click **Actions**.
- In the left sidebar, click the workflow you want to see.



Summary

After you set up everything above, you will end up with a repo like this:



You can begin your adventure from here. Good luck, have fun!

References

- Maven - How to create a Java project Mkyong.com
- Maven Archetype Plugin – archetype:generate
- Maven – Introduction to the POM
- Maven – Maven in 5 Minutes
- .gitignore file - ignoring files in Git | Atlassian Git Tutorial
- Quickstart for GitHub Actions - GitHub Docs