Grouper Development Environment using Gradle

Overview

A functional gradle build environment exists in the Github repository in branch "gradle". The build is orchestrated by a parent build.gradle file for the root grouper project and a number of individual build.gradle file for every module that Grouper presently houses. Each build.gradle file inherits from the parent build.gradle file and in turn is able to specify its own requirements for dependency management and source tree structure. The list of Grouper modules that are recognized by the Gradle build is specified via a settings.gradle in a specific order to account for the build dependencies of the modules. There is also one gradle.properties file which hosts specific build settings for the project, such as groupid, java version and dependency versions used throughout the project.

Build

To run the full build, issue the following command at the root:

```
./gradlew build -x test -x javadoc --parallel --stacktrace
```

The same command can be used inside a specific module directory to build that module only.

Branch Info

You can use "-x" to skip specific tasks during the build lifecycle.

The above command will use the Gradle wrapper to download an appropriate Gradle version once to be used for the build. This removes the requirement for a developer to explicitly have Gradle downloaded and installed, but if a Gradle installation is already available, the same command can be used via invoking the native Gradle instance installed. Build artifacts are produced for each module inside the build/libs directory. JAR files are produced for main, tests and javadoc.

```
./gradle build -x test -x javadoc --parallel --stacktrace
```

Windows Gradle Wrapper

Windows version of the wrapper command is also available as gradlew.bat

The following gradle commands may also prove useful during development:

```
# Show all the project dependencies
./gradle dependencies

# Show where a specific dependency for a gradle configuration comes from
./gradle dependencyInsight --dependency <name> --configuration <compile/runtime/testCompile/...>

# Prepare an Intellij IDEA workspace for Grouper
./gradle idea

# Prepare an Eclipse project workspace for Grouper
./gradle eclipse
```

Gradle Wrapper Updates

The Gradle wrapper need only exist at the root of the project. The wrapper can be updated to note a specific newer version of a Gradle via the following command:

```
./gradle wrapper
```

Javadocs

To produce the Javadoc for the entire project, run the following command at the root:

```
./gradlew build alljavadoc -x test -x javadoc --parallel --stacktrace
```

The above command aggregates the javadocs for all grouper modules inside the root build/javadoc directory.

Travis CI Builds

Grouper is configured against Travis CI to execute the full build on every commit. Travis builds all grouper branches provided there is a .travis.yml inside the branch at the root of the repository. This is a YAML configuration file that tells Travis what version of Java should be available, which commands to use for the build lifecycle and which scripts to execute after a successful build.

> **Branch Info**
>
> The configuration file must be massaged for every branch to tell Travis to limit the build to that branch only, for faster builds.

Publishing Javadocs

As part of the Travis CI build, Javadocs are published to the gh-pages of the repository, available on the web here, once a successful builds. This is handled via the javadocs-ghpages.sh script that runs after the build. The script will attempt to collect all project javadocs and push them to the appropriate branch. In order for the push to succeed, an OAUTH token is made available to the script to do the job. The OAUTH token must be generated by a developer with push rights to the repo, and must be encrypted and placed inside the .travis.yml file under a "secure" heading.

> **Branch Info**
>
> The script file must be massaged for every branch to tell Travis to limit the build to that branch only.

OAUTH token can be encrypted for Travis via the following command:

```
gem install travis
travis login --github-token <TOKEN>
travis encrypt GH_TOKEN=<GH-TOKEN>-r Internet2/grouper
```

- The TOKEN must be generated by you inside your Github profile settings. This is used to allow you to log into Travis.
- The GH-TOKEN must be generated by you inside your Github profile settings. This is used to allow Travis to push to the gh-pages branch.
- When the values are encrypted, they are printed back on the screen. Copy/paste them to the .travis.yml file under env/secure entry.

Publishing SNAPSHOTs

As part of the Travis CI build, SNAPSHOT artifacts for all grouper modules listed in the settings.gradle file are published to the Sonatype snapshot repositories for Grouper once a successful builds. This is handled via the deploy-to-sonatype.sh script that runs after the build. The script will attempt will collect all JAR artifacts produced by the build and will attempt to publish them. The command that used by the script is:

```
./gradlew uploadArchives -DpublishSnapshots=true -DsonatypeUsername=$SONATYPE_USER -
DsonatypePassword=$SONATYPE_PWD
```

The above command invokes the "uploadArchives" task of Gradle to start publishing. The configuration of this task is specified in the root build.gradle file.

In order for the command to succeed, a developer must provide his/her own Sonatype credentials, and must also be authorized to publish artifacts to the edu.internet2.grouper namespace. Credentials that are required for the command to succeed are encrypted and placed inside the .travis.yml file under a "secure" heading. The developer must also be authorized to have access to the "Settings" area of the Grouper github repository.

> **Branch Info**
>
> The script file must be massaged for every branch to tell Travis to limit the build to that branch only.

Sonatype credentials can be encrypted for Travis via the following command:

```
gem install travis
travis login --github-token <TOKEN>
travis encrypt SONATYPE_USER=yourUsername -r Internet2/grouper
travis encrypt SONATYPE_PWD=yourPassword -r Internet2/Grouper
```

- The TOKEN must be generated by you inside your Github profile settings.
- When the values are encrypted, they are printed back on the screen. Copy/paste them to the .travis.yml file under env/secure entry.

Development environment

- Open morphstring in eclipse
- Delete .classpath and .project
- Set java home and path

```
C:\Users\mchyzer\Documents\GitHub\grouper\grouper>set JAVA_HOME=C:\dev_inst\java7
C:\Users\mchyzer\Documents\GitHub\grouper\grouper>set PATH=%JAVA_HOME%\bin;%PATH%
```

- sdf