Spring Framework - Eclipse/STS Project Import Guide

This document will guide you through the process of importing the Spring Framework projects into Eclipse or the Spring Tool Suite (*STS*). It is recommended that you have a recent version of Eclipse. As a bare minimum you will need Eclipse with full Java 17 support and Eclipse Buildship.

The following instructions have been tested against STS 4.12.0 (download) (based on Eclipse 4.21) with Eclipse Buildship. The instructions should work with the latest Eclipse distribution as long as you install Buildship. Note that STS 4 comes with Buildship preinstalled.

If you are using Eclipse 4.21, you will need to install Java 17 Support for Eclipse 2021-09 (4.21) from the Eclipse Marketplace.

Steps

When instructed to execute `./gradlew` from the command line, be sure to execute it within your locally cloned `spring-framework` working directory.

1. Ensure that the *Forbidden reference (access rule)* in Eclipse is set to `Info` (Preferences → Java → Compiler → Errors/Warnings → Deprecated and restricted API → Forbidden reference (access rule)).
2. Optionally install the Kotlin Plugin for Eclipse if you need to execute Kotlin-based tests or develop Kotlin extensions.
    - **NOTE**: As of September 21, 2021, it appears that the Kotlin Plugin for Eclipse does not yet work with Eclipse 4.21.
3. Optionally install the AspectJ Development Tools (*AJDT*) if you need to work with the `spring-aspects` project.
    - **NOTE**: As of September 21, 2021, it appears that the AspectJ Development Tools do not yet work with Eclipse 4.21.
4. Optionally install the TestNG plugin in Eclipse if you need to execute individual TestNG test classes or tests in the `spring-test` module.
    - As an alternative to installing the TestNG plugin, you can execute the `org.springframework.test.context.testng.TestNGTestSuite` class as a "JUnit 5" test class in Eclipse.
5. Build `spring-oxm` from the command line with `./gradlew :spring-oxm:check`.
6. To apply Spring Framework specific settings, run `./gradlew cleanEclipse eclipse` from the command line.
7. Import all projects into Eclipse (File → Import → Gradle → Existing Gradle Project → Navigate to the locally cloned `spring-framework` directory → Select Finish).
    - If you have not installed AJDT, exclude the `spring-aspects` project from the import, if prompted, or close it after the import.
    - If you run into errors during the import, you may need to set the *Java home* for Gradle Buildship to the location of your JDK 8 installation in Eclipse (Preferences → Gradle → Java home).
8. If you need to execute JAXB-related tests in the `spring-oxm` project and wish to have the generated sources available, add the `build/generated-sources/jaxb` folder to the build path (right click on the `jaxb` folder and select "Build Path → Use as Source Folder").
    - If you do not see the `build` folder in the `spring-oxm` project, ensure that the "Gradle build folder" is not filtered out from the view. This setting is available under "Filters" in the configuration of the Package Explorer (available by clicking on the *three vertical dots* in the upper right corner of the Package Explorer).

9. Code away!

Known Issues

1. `spring-core` should be pre-compiled due to repackaged dependencies.
   - See `*RepackJar` tasks in the `spring-core.gradle` build file.
2. `spring-oxm` should be pre-compiled due to JAXB types generated for tests.
   - Note that executing `./gradlew :spring-oxm:check` as explained in the *Steps* above will compile `spring-core` and generate JAXB types for `spring-oxm`.
3. `spring-aspects` does not compile due to references to aspect types unknown to Eclipse.
   - If you installed *AJDT* into Eclipse it should work.
4. While JUnit tests pass from the command line with Gradle, some may fail when run from the IDE.
   - Resolving this is a work in progress.
   - If attempting to run all JUnit tests from within the IDE, you may need to set the following VM options to avoid out of memory errors: `-XX:MaxPermSize=2048m -Xmx2048m -XX:MaxHeapSize=2048m`

Tips

In any case, please do not check in your own generated `.classpath` file, `.project` file, or `.settings` folder. You'll notice these files are already intentionally in `.gitignore`. The same policy holds for IDEA metadata.