Prerequisite Knowledge

To be successful with the Ignition Module SDK, you'll want to become acquainted with the following technologies and concepts *before* you get started.

Java SE

Java is the language that Ignition is written in, and the language that modules must also be written in. Therefore, to even get started with the SDK, you'll need to have a decent foundation in the Java language and framework. Concepts such as package management, classpaths, and JAR files are frequently used in the development of Ignition modules. Ignition is built against Java 8.

See also: Oracle's Java Developer Portal

Build Systems

New in 7.7.2 Ignition SDK!

For Ignition 7.7.6+, we've completely revamped how the Ignition SDK works to minimize friction and let developers focus on building tools for modern automation! No need to download a bunch of *jars* and create Ant tasks. The modernized SDK is much smaller, creates modules from your source and dependencies with our **Ignition Maven Plugin.** We've added a public Maven artifact repository for dependency management to help you avoid classpath/dependency hell and remove some of the friction of getting started. We've hosted a number of example projects on our Github site and encourage people to clone, fork or even send pull requests if you have something to add. See the SDK_examples section on how to clone, build and deploy an example project in minutes!

Ant is the build file system used by past versions of Ignition's SDK. While not strictly necessary for module development, understanding and using Ant for those versions will make things much easier. Each example project in the legacy SDK (which includes SDK versions up through Ignition 7.7.6) includes an Ant build file `build.xml` which can be modified and adapted as necessary.

Current versions of the Ignition SDK use **Gradle** instead of Ant. Gradle is often easier for Java developers to understand, but you can use any Java build tools you are comfortable with.

Maven is the predominant dependency management tool utilized in Java, and is also one of the most popular build systems. It's 'convention over configuration' approach has laid the foundation for how modern Java applications are structured. If you use our Ignition Maven Plugin, you will need to be familiar with `pom.xml` files and the Maven build cycle.

See also:

Apache Ant Homepage Maven Homepage Gradle Homepage

Ignition

Of course, to build an Ignition module, it's helpful to have good background in working with Ignition itself. You should have a good understanding of how many of the platform services work and are configured through the gateway, such as database connections, authentication profiles, SQLTags providers, OPC connections, and projects. For some targeted project types, such as a device driver for OPC-UA, a wide knowledge of the platform isn't necessary, but you should still be acquainted with how the existing drivers are configured and used.