Dokka

Dokka is a documentation engine for Kotlin, performing the same function as javadoc for Java. Just like Kotlin itself, Dokka fully supports mixed-language Java/Kotlin projects. It understands standard Javadoc comments in Java files and KDoc comments in Kotlin files, and can generate documentation in multiple formats including standard Javadoc, HTML and Markdown.

Using Dokka

Full documentation is available at https://kotlin.github.io/dokka/1.5.0/

Using the Gradle plugin

Note: If you are upgrading from 0.10.x to a current release of Dokka, please have a look at our migration guide

The preferred way is to use `plugins` block.

build.gradle.kts:

```
plugins {
    id("org.jetbrains.dokka") version "1.5.0"
}

repositories {
    mavenCentral()
}
```

The plugin adds `dokkaHtml`, `dokkaJavadoc`, `dokkaGfm` and `dokkaJekyll` tasks to the project.

Applying plugins

Dokka plugin creates Gradle configuration for each output format in the form of `dokka${format}Plugin`:

```
dependencies {
    dokkaHtmlPlugin("org.jetbrains.dokka:kotlin-as-java-plugin:1.5.0")
}
```

You can also create a custom Dokka task and add plugins directly inside:

```
val customDokkaTask by creating(DokkaTask::class) {
    dependencies {
        plugins("org.jetbrains.dokka:kotlin-as-java-plugin:1.5.0")
    }
}
```

Please note that `dokkaJavadoc` task will properly document only single `jvm` source set

To generate the documentation, use the appropriate `dokka${format}` Gradle task:

```
./gradlew dokkaHtml
```

Please see the Dokka Gradle example project for an example.

We encourage users to create their own plugins and share them with the community on official plugins list.

Android

Make sure you apply Dokka after `com.android.library` and `kotlin-android`.

README.md

```
buildscript {
    dependencies {
```

```
        classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:${kotlin_version}")
        classpath("org.jetbrains.dokka:dokka-gradle-plugin:${dokka_version}")
    }
}
repositories {
    mavenCentral()
}
apply(plugin= "com.android.library")
apply(plugin= "kotlin-android")
apply(plugin= "org.jetbrains.dokka")


dokkaHtml.configure {
    dokkaSourceSets {
        named("main") {
            noAndroidSdkLink.set(false)
        }
    }
}
```

Multi-module projects

For documenting Gradle multi-module projects, you can use `dokka${format}Multimodule` tasks.

```
tasks.dokkaHtmlMultiModule.configure {
    outputDirectory.set(buildDir.resolve("dokkaCustomMultiModuleOutput"))
}
```

`DokkaMultiModule` depends on all Dokka tasks in the subprojects, runs them, and creates a toplevel page with links to all generated (sub)documentations

Using the Maven plugin

The Maven plugin does not support multi-platform projects.

Documentation is by default generated in `target/dokka`.

The following goals are provided by the plugin:

- `dokka:dokka` - generate HTML documentation in Dokka format (showing declarations in Kotlin syntax)
- `dokka:javadoc` - generate HTML documentation in Javadoc format (showing declarations in Java syntax)
- `dokka:javadocJar` - generate a .jar file with Javadoc format documentation

Applying plugins

You can add plugins inside the `dokkaPlugins` block:

```
<plugin>
    <groupId>org.jetbrains.dokka</groupId>
    <artifactId>dokka-maven-plugin</artifactId>
    <version>${dokka.version}</version>
    <executions>
        <execution>
            <phase>pre-site</phase>
            <goals>
                <goal>dokka</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <dokkaPlugins>
            <plugin>
                <groupId>org.jetbrains.dokka</groupId>
                <artifactId>kotlin-as-java-plugin</artifactId>
                <version>${dokka.version}</version>
            </plugin>
        </dokkaPlugins>
    </configuration>
</plugin>
```

Please see the Dokka Maven example project for an example.

Using the Command Line

To run Dokka from the command line, download the Dokka CLI runner. To generate documentation, run the following command:

```
java -jar dokka-cli.jar <arguments>
```

You can also use a JSON file with dokka configuration:

```
java -jar <dokka_cli.jar> <path_to_config.json>
```

Output formats

Dokka documents Java classes as seen in Kotlin by default, with javadoc format being the only exception.

- `html` - HTML format used by default
- `javadoc` - looks like JDK's Javadoc, Kotlin classes are translated to Java
- `gfm` - GitHub flavored markdown

- `jekyll` - Jekyll compatible markdown

If you want to generate the documentation as seen from Java perspective, you can add the `kotlin-as-java` plugin to the Dokka plugins classpath, eg. in Gradle:

```
dependencies{
    implementation("...")
    dokkaGfmPlugin("org.jetbrains.dokka:kotlin-as-java-plugin:${dokka-version}")
}
```

FAQ

If you encounter any problems, please see the FAQ.

Releases 40

1.5.0 Alpha Latest
3 days ago

+ 39 releases

Packages

No packages published

Contributors 78

+ 67 contributors

Languages

Kotlin 96.9% CSS 1.2% JavaScript 0.8% TypeScript 0.8% SCSS 0.2% Java 0.1%