Internal Dependencies

References

- Analyze java package metrics in a graph database
- Calculate metrics
- Neo4j Python Driver

1 - Modules

List the modules this notebook is based on. Different sorting variations help finding modules by their features and support larger code bases where the list of all modules gets very long.

Only the top 30 entries are shown. The whole table can be found in the following CSV report: `List_all_Typescript_modules`

Table 1a - Top 30 modules with the highest element count

	moduleName	numberOfElements	incomingDependencies	outgoingDependencies	moduleFullQualifiedName
0	react-router-dom	35	2	148	/home/runner/work/code-graph-analysis-pipeline...
1	react-router-native	12	0	24	/home/runner/work/code-graph-analysis-pipeline...
2	react-router	6	19	17	/home/runner/work/code-graph-analysis-pipeline...
3	server	6	0	40	/home/runner/work/code-graph-analysis-pipeline...

Table 1b - Top 30 modules with the highest number of incoming dependencies

The following table lists the top 30 internal modules that are used the most by other modules (highest count of incoming dependencies, highest in-degree).

	moduleName	numberOfElements	incomingDependencies	outgoingDependencies	moduleFullQualifiedName
0	react-router	6	19	17	/home/runner/work/code-graph-analysis-pipeline...
1	react-router-dom	35	2	148	/home/runner/work/code-graph-analysis-pipeline...
2	react-router-native	12	0	24	/home/runner/work/code-graph-analysis-pipeline...
3	server	6	0	40	/home/runner/work/code-graph-analysis-pipeline...

Table 1c - Top 30 modules with the highest number of outgoing

dependencies

The following table lists the top 30 internal modules that are depending on the highest number of other modules (highest count of outgoing dependencies, highest out-degree).

	moduleName	numberOfElements	incomingDependencies	outgoingDependencies	moduleFullQualifiedName
0	react-router-dom	35	2	148	/home/runner/work/code-graph-analysis-pipeline...
1	server	6	0	40	/home/runner/work/code-graph-analysis-pipeline...
2	react-router-native	12	0	24	/home/runner/work/code-graph-analysis-pipeline...
3	react-router	6	19	17	/home/runner/work/code-graph-analysis-pipeline...

Table 1d - Top 30 modules with the lowest element count

	moduleName	numberOfElements	incomingDependencies	outgoingDependencies	moduleFullQualifiedName
0	react-router	6	19	17	/home/runner/work/code-graph-analysis-pipeline...
1	server	6	0	40	/home/runner/work/code-graph-analysis-pipeline...
2	react-router-native	12	0	24	/home/runner/work/code-graph-analysis-pipeline...
3	react-router-dom	35	2	148	/home/runner/work/code-graph-analysis-pipeline...

Table 1e - Top 30 modules with the lowest number of incoming dependencies

The following table lists the top 30 internal modules that are used the least by other modules (lowest count of incoming dependencies, lowest in-degree).

	moduleName	numberOfElements	incomingDependencies	outgoingDependencies	moduleFullQualifiedName
0	react-router-native	12	0	24	/home/runner/work/code-graph-analysis-pipeline...
1	server	6	0	40	/home/runner/work/code-graph-analysis-pipeline...
2	react-router-dom	35	2	148	/home/runner/work/code-graph-analysis-pipeline...
3	react-router	6	19	17	/home/runner/work/code-graph-analysis-pipeline...

Table 1f - Top 30 modules with the lowest number of outgoing dependencies

The following table lists the top 30 internal modules that are depending on the lowest number of other modules (lowest count of outgoing dependencies, lowest out-degree).

	moduleName	numberOfElements	incomingDependencies	outgoingDependencies	moduleFullQualifiedName
0	react-router	6	19	17	/home/runner/work/code-graph-analysis-pipeline...
1	react-router-native	12	0	24	/home/runner/work/code-graph-analysis-pipeline...
2	server	6	0	40	/home/runner/work/code-graph-analysis-pipeline...
3	react-router-dom	35	2	148	/home/runner/work/code-graph-analysis-pipeline...

2 - Cyclic Dependencies

Cyclic dependencies occur when one module uses an elements of another module and vice versa. These dependencies can lead to problems when one of these modules needs to be changed.

Table 2a - Cyclic Dependencies Overview

Show the top 40 cyclic dependencies sorted by the most promising to resolve first. This is done by calculating the number of forward dependencies (first cycle participant to second cycle participant) in relation to backward dependencies (second cycle participant back to first cycle participant). The higher this rate (approaching 1), the easier it should be to resolve the cycle by focussing on the few backward dependencies.

Only the top 40 entries are shown. The whole table can be found in the following CSV report: `Cyclic_Dependencies_for_Typescript`

Columns:

- *projectFileName* identifies the project of the first participant of the cycle
- *modulePathName* identifies the module of the first participant of the cycle
- *dependentProjectFileName* identifies the project of the second participant of the cycle
- *dependentModulePathName* identifies the module of the second participant of the cycle
- *forwardToBackwardBalance* is between 0 and 1. High for many forward and few backward dependencies.
- *numberForward* contains the number of dependencies from the first participant of the cycle to the second one
- *numberBackward* contains the number of dependencies from the second participant of the cycle back to the first one
- *someForwardDependencies* lists some forward dependencies in the text format "type1 -> type2"
- *backwardDependencies* lists the backward dependencies in the format "type1 <- type2" that are recommended to get resolved

projectFileName	moduleName	dependentProjectFileName	dependentModulePathName	forwardToBackwardBalance	numberForward	numberBackward	forwardDep

Table 2b - Cyclic Dependencies Break Down

Lists modules with cyclic dependencies with every dependency in a separate row sorted by the most promising dependency first.

Only the top 40 entries are shown. The whole table can be found in the following CSV report: `Cyclic_Dependencies_Breakdown_for_Typescript`

Columns in addition to Table 2a:

- *dependency* shows the cycle dependency in the text format "type1 -> type2" (forward) or "type2<- type1" (backward)

projectFileName	moduleName	dependentProjectFileName	dependentModulePathName	dependency	forwardToBackwardBalance	numberForward	numberBackwar

Table 2c - Cyclic Dependencies Break Down - Backward Dependencies Only

Lists modules with cyclic dependencies with every dependency in a separate row sorted by the most promising dependency first. This table only contains the backward dependencies from the second participant of the cycle back to the first one that are the most promising to resolve.

Only the top 40 entries are shown. The whole table can be found in the following CSV report: `Cyclic_Dependencies_Breakdown_BackwardOnly_for_Typescript`

projectFileName	moduleName	dependentProjectFileName	dependentModulePathName	dependency	forwardToBackwardBalance	numberForward	numberBackward

3 - Module Usage

Table 3a - Elements that are used by multiple modules

This table shows the top 40 modules that are used by the highest number of different modules. The whole table can be found in the CSV report `WidelyUsedTypescriptElements`.

fullQualifiedDependentElementName	dependentElementModuleName	dependentElementName	dependentElementLabels	numberOfUsingModules

Table 3b - Elements that are used by multiple modules

This table shows the top 30 modules that only use a few (compared to all existing) elements of another module. The whole table can be found in the CSV report `ModuleElementsUsageTypescript`.

sourceModuleName	dependentModuleName	dependentElementsCount	dependentModuleElementsCount	elementUsagePercentage	dependentElementFullNameExam

Table 3c - Distance distribution between dependent files

This table shows the file directory distance distribution between dependent files. Intuitively, the distance is given by the fewest number of change directory commands needed to navigate between a file and a dependency it uses. Those are aggregate to see how many dependent files are in the same directory, how many are just one change directory command apart, and so on.

	dependency.fileDistanceAsFewestChangeDirectoryCommands	numberOfDependencies	numberOfDependencyUsers	numberOfDependencyProviders	examples
0	0	1	1	1	[./server.tsx uses ./index.tsx]
1	4	6	4	2	[./index.tsx uses ./index.ts, ./index.tsx uses...