Structure API Basics

Classes to Start With

Use ...	to ...
StructureManager *and* Forest	create and manipulate structures — add issues to the hierarchy, move issues within hierarchy, delete issues from the hierarchy, listen to structure events

The API uses an open source library Integers, which provides primitive-type collections with `java.util`-like interfaces. When working with `Forest`, you will typically use `LongList` and `LongArray` (an implementation of `LongList`).

See API Usage Samples to get the idea how to work with those interfaces.

The dependency on the Integers library is added automatically when your add dependency on the API. Same goes for another dependency on the small JetBrains annotations library, that provides `@Nullable` and `@NotNull` annotations.

More Power

Use ...	to ...
StructureViewManager	create and manipulate structure views
StructureSynchronizer	create a new synchronizer and declare it in a Synchronizer Module
StructureSyncManager	manage synchronizers
StructureBackupManager	backup structure to a file and restore it back
StructureJobManager	schedule asynchronous jobs
IssueEventBridge	listen for aggregated issue events
StructureQuery	to search for issues added to a structure with hierarchical constraints, as in `structure()` JQL function
Aggregate and AggregateCalculator	calculate aggregate values, such as Total Time Spent or Progress
Aggregates	obtain aggregate instances for summing up time tracking fields or votes
ProgressAggregateFactory	obtain aggregate instances for calculating issue progress (as shown by the Progress column)
DoubleSum and NumericCustomField	create an aggregate instance for summing up a numeric custom field

Next: consider Controlling Compatibility and Making Structure Dependency Optional