Importer API: Mapping

Once one or more Alf model unit files (and their subunits, if any) have been parsed, you can use the *AlfImporter.compile* method to constraint check all cached units and, if there are no constraint violations, map them into the UML model. If there are constraint violations, then these are reported in the MagicDraw Message Window. The *compile* method returns a Boolean indicating whether constraint checking was successful or not. You can also check the success of the last compilation by calling the *isSuccessful* operation.

The *compile* method takes a Java Path as a parameter, but this is only used to update the *ProgressStatus* object provided when the AlfImporter was created (if any), and it may be null (in which case the progress status is not updated). Since mapping results in an update to the UML model, the *compile* method should be called within a MagicDraw session. This can be done conveniently using the *AlfActionUtil.executeSession* method, which also ensures that any automatic Alf compilation is turned off during the session, so new compilations are not accidentally triggered by updates from the mapping process.

> ⚠ Using an *AlfImporter* in a Project is incompatible with subsequently using an *AlfCompiler* in that Project. Once you have completed an importation, you must use the *AlfActionUtil,.resetActiveProject* method to return the currently active project to a state in which the *AlfCompiler* can be used, or use to *AlfActionUtil.resetProject* method to similarly reset a specific project. You should do this even if the importation was not successful. Alternatively, you can use the *AlfActionUtil.importFrom* method, which allows for subsequent Alf compilation after importation (see Importer API: Utilities).

```
AlfImporter importer = new AlfImporter(modelDirectory, progressStatus);
Path path = Paths.get(modelDirectory, modelFileName);
if (importer.parse(path)) {
    AlfActionUtil.executeSession("Import Alf", new Runnable() {
        public void run() {
            importer.compile(path);
        }
    });
}
// It is unsafe to use the AlfCompiler API at this point.

AlfActionUtil.resetActiveProject();

// It is safe to use the AlfCompiler API from here on...
```

Rather than separately parsing and then compiling imported Alf files, you can do both with one call to the *importFile* method. Given a Java *Path* object for a single file in the model directory identified when the *AlfImporter* object was created, the *importFile* method parses the model unit in the given file (and its subunits, if any), performs constraint checking and, if that is all successful, compiles the unit into the UML model. The method returns a Boolean indicating whether the import was successful or not. If not, any syntactic errors or constraint violations are reported in the MagicDraw Message Window. As for the *compile* method, the *importFile* method should be called within a MagicDraw session, since it may potentially update the UML model if a mapping is carried out.

```
AlfImporter importer = new AlfImporter(modelDirectory, progressStatus);
AlfActionUtil.executeSession("Import Alf", new Runnable() {
    public void run() {
        importer.importFile(Paths.get(modelDirectory,
modelFileName));
    }
});
AlfActionUtil.resetActiveProject();
```

Related Pages

- Importer API: Parsing