Building and cleaning

As discussed in Dependency Management, when you make changes within a model, the Alf compiler normally automatically processes any Alf code that might be affected by these changes. However, in some cases the Alf compiler may not have enough information to realize that a change in the model will correct an error in previously erroneous Alf code. Or, you may have turned off the **Build Automatically** environment option, in which case the Alf compiler will mark elements needing recompilation without automatically recompiling them.

For cases such as the above, the Alf plugin provides two commands that allow you to manually trigger the recompilation of Alf code, which is known as doing an Alf *build* of your project. These commands are available by selecting **Tools > Alf** from the main menu.

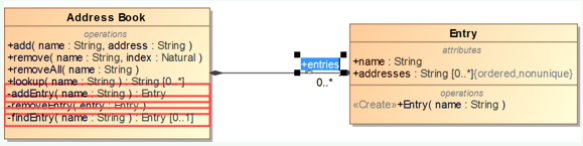
- **Build Project** – A *normal build* triggers the recompilation of the Alf bodies of all model elements marked as having Alf compilation errors or otherwise needing to be recompiled.
- **Clean Project** – A *clean build* triggers the recompilation of all Alf code in your project, whether it is marked as needing recompilation or not.

Build Project

To initiate a normal build of your project, select **Tools > Alf > Build Project**. If there are any elements in your project that are marked with either an Alf compilation error annotation or an Alf recompilation-needed warning annotation (see Dependency Management), then the Alf bodies for these elements will be recompiled. Any previously erroneous Alf code that compiles successfully has its error annotation removed, while any code with new compilation errors has a new error annotation added. All recompilation-needed warning annotations are removed.

A progress bar will appear to track the progress of the build. If a build takes a long time, then you can cancel it before it is complete. If you cancel a build, then any compilation of Alf code that happened before the cancellation will be reflected in your model. Elements that have not yet been processed will still have the annotations that they had before the build was initiated. To process those remaining elements, initiate a new build.

The figure below shows the error annotations caused by deleting the Association in the Address Book example and then recreating it as a new Association, so the error annotations remain (see also the discussion of this example in Dependency Management). Selecting **Tools > Alf > Build Project** initiates an Alf build of the project, resulting in the successful recompilation of the indicated operations and the removal of the error annotations.



Example showing the need to manually rebuild

Clean Project

To initiate a clean build of your project, select **Tools > Alf > Clean Project**. This will result in the removal of all existing Alf annotations in your project and the recompilation of all Alf code. The Alf compiler will then add error annotations to any elements with Alf code that has compilation errors.

Compilation of template bindings in your Alf code result in the generation of instantiated Classes in the *$$Template Bindings* Package in your project. Over time, Classes in this Package may accumulate that are no longer necessary for your project. A clean rebuild first cleans out this Package, so, at the end of the build, it contains only the instantiations truly needed for the Alf code actually in your project. If your Alf code no longer has any template bindings, then the Package is removed entirely from your model.

If you move a lot of the elements in a project around between Packages, the Alf compiler may lose track of some potential dependencies and not properly mark all necessary recompilations. Therefore, if you do significant restructuring of your model, it is a good idea to then perform a clean rebuild of all the Alf code, which will ensure all the code is checked for errors and that the dependency tracking is properly rebuilt.

Table of Contents

Related pages

- The Alf compiler
- Dependency Management
- Environment options

If your project contains a lot of Alf code, then a clean rebuild will likely take quite a bit longer than a normal build. A progress bar will appear to track the progress of the build and allow you to cancel it before it completes. However, if you cancel a clean build before it completes, then the model will be updated with the results of only the compilations that occurred before the cancellation, which can result in inconsistencies. To avoid this, you can perform an **Undo** to undo all changes made by the canceled build.