Stubs

Suppose we want to describe one of the existing languages inside of MPS (for example, we can roughly consider baseLanguage as a description of Java in MPS). In this case, the libraries written for this language should be accessible from MPS. The stubs a spect of a language helps to create stub models for such libraries. These models are used to reference the code outside of MPS.

Stubs Language

There are two things that can be described using stubs aspect
1. The StubCreator describes how to parse existing code and convert it into models
2. The StubSolutions describe standard libraries for the language and how MPS can find them on the user's machine (e.g. for baseLanguage, there are 2 stub solutions - JDK for JDK classes and MPS.Classpath for MPS classes)

Stub Creator

name	Any descriptive name. Note that this name should not be changed after this stub manager is published, otherwise your users will get broken libraries in their modules
modelDescriptors	Should return modelDescriptors for models that can be loaded from the given location. Do not parse models here
updateModel	The given model should be updated using data from given location. This means that all entities from this location should be added to the models.
rootNodeDescriptors	This is used for "Go to Node" action to find the names of root nodes quickly, without loading a model

Stub Solution

module	Module to add stub models to. The module can be created by "ctrl-space -> create new..." or via "Language Properties -> Design Time -> StubSolutions"
creator	Stub Creator to be used to load model of this solution
roots	Should return a list of paths to load models from

Note that you can't re-create a stub solution after you have published it (even with the same name!). This happens because these solutions are referenced from user modules by automatically generated moduleId, which can't be preserved between deletion and creation.

Note that there's no need for using stub solutions for libraries that you distribute together with your language. Such libraries can be simply added as runtime or design-time libraries in language properties. Stub solutions are only applicable when paths to load stubs from can't be statically specified (like JDK paths, which can be different on different computers)

Usage

Let's consider how stubs are used.

Author of the language does the following things:
1. Creates StubCreator that can load stub models for his language
2. Creates StubSolutions describing standard libraries for his language (the libraries that are pre-installed on the user's machine and are not included into the language's packaging)

After the things mentioned are done, any user of this language can
1. Use the described libraries directly (import stub models, view and reference them, etc.)
2. Use his own libraries that can be loaded by the created StubCreator.

Let's see how to add a new library of a new type to solution:
1. Go to Solution Properties -> External code -> Libraries
2. Add a new library (select a path where it is situated)
3. Choose an appropriate manager for this library

Performance

For now, stubs are reloaded on classes reload.
Stubs are invalidated and loaded lazily. This means that

- the contents of stub model are not loaded before they are queried
- if neither source files, nor the stub manager for some stub model were changed since the last reload, the model won't be reloaded

Despite the optimizations considered, it is better to mention that there are still some bottlenecks.
1. The list of models is queried every time the classes are reloaded, which means that modelDescriptors block of Stub Manager is better to execute fast.
2. As the aim of stub models is to reference the outside code and help to compute types, there's no need to include anything non-referenceable into stub model. E.g. nothing inside Java method can be referenced from outside the method, so there's no need to include method bodies into stub models. This will only make the loading slower and the models bigger.

Warning: Never set stubs containing same models in different modules. Most probably, this won't work.

Previous Next