Multi Language Support (MLS) – User perspective

Position paper for Eclipse Languages Workshop

Guy Harpaz

The Eclipse Platform is the standard development environment for Java development, but although there are open source and commercial plugins for almost any development language, it is not yet the standard environment for other languages.

As I see it, the main reason for it is absence of multi language support. The problem is even more severe when the subject of dynamic languages is raised.

Multi Language Support (MLS) in Eclipse can be divided to three levels:

1. The first level is Eclipse as an IDE for a single specific language. The developer can download Eclipse as an IDE for the one language he is using.

2. The second level of MLS in Eclipse is for the "multi language developer". A developer that is developing separately in few different languages.

3. The last level of MLS, and the most complex one, is very similar to the second level, but here the developer is not just developing in more then one language but he is doing it in the same file.

As I see it, the first level of MLS does not raise any user's demands (except of being a good IDE). A single language user cares only for the environment he is using and is indifferent to the level of integration with other languages.

In the second level of MLS users expect Eclipse as an ML supporting IDE to be familiar when they are switching between languages. Although users use different plugins, the level of standardization should be high.

The most demanding level of MLS is level three, in which a user using several languages within the same file expects a standardized environment. This is not a farfetched example but a rather common task to a Web developer using HTML, Java Script, PHP, and CSS in the same file.
By standardization I mean that users want a consistent environment as oppose to plugins patchwork. Users obviously prefer a complete environment built for their needs.

Following is one example to a basic feature that is commonly used but is not standardized at the current version.

By examining the global preference tree of Eclipse one can immediately see that setting up a basic feature such as syntax coloring can be a rather complicated task for the Multi Language programmer.
For example: A Web developer who uses HTML, Java Script, PHP & CSS in the same file and wants to change syntax coloring can do so only by editing the preferences in 4 (=No. of languages used) different pages of the "preference tree".
Instead of changing preferences in different pages, preferences of the editor should be generalized. A feature such as syntax coloring, which exists in every textual language, should be controlled from one location.

There are many technical challenges in MLS, some of them are very interesting, but I think we should first examine the MLS from the user perspective and try to meet users' demands.