Staying Pragmatic

Position Paper for Eclipse Languages Workshop

Mike Milinkovich
October 14, 2005

One of the immediate issues that surfaces when discussing this topic is that no two people mean the same thing when they refer to “multi-language support” in Eclipse.

Just a partial list of how the term has been interpreted includes:

- Some plug-in providers wish that the APIs for different language tools were identical so that their tools can work with (say) both CDT and JDT with no porting on their part.

- To some, it means the ability to create a new language IDE by very simple steps. Perhaps even as simple as running a wizard that ask that you specify the grammar, compiler, build tool, etc. for a particular language and an entry-level IDE is created for you.

- Some define the issue as building base-level frameworks to help accelerate the work of those who want to develop editors, debuggers and build systems for a specific language. Proponents of this view are sometimes simply looking for the ability to create a new language IDE by extending pre-existing frameworks as opposed to copying code.

Another issue is how to define the solution. Some focus on how to simplify the creation of new tools. Others are interested in seeing more effective code re-use across the existing projects such as JDT and CDT.

In a probable fit of wild fantasy, I am hoping that the workshop will have two major outcomes:

1. A lexicon that we can work with to describe the different scenarios. “Multi-language support” is too broad and vague to be useful in conversation.

2. A set of pragmatic goals on what could be accomplished by the Eclipse community over the next couple of release cycles. This will likely include a list of modifications (hopefully modest) to the platform (including LTK and JDT), along with project ideas for future work.