Conversion Tools - Feature #3883

eclipse plug for developing 4GL code using FWD including editing, syntax checks, running conversion

01/15/2019 04:52 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Conversion Tools - Feature #6319: IntelliJ plugin	**New**
Related to Conversion Tools - Feature #1757: update ANTLR to latest version	**New**

History

#1 - 01/15/2019 04:52 PM - Greg Shah

Eclipse plugin for integration with syntax check and running conversion (et al). This must also provide the regular Eclipse support for 4GL editor coloring, syntax expansion, source debugging and so forth). A developer must be able to continue 4GL development using FWD only (no Progress Developer Studio which is a licensed product).

#2 - 04/11/2021 12:21 PM - Greg Shah

There is an Eclipse "simple" plugin framework called XText. It is probably not suitable for our usage. See this answer about integrating an external parser into Eclipse using Xtext. XText is essentially a framework in which you plugin a bunch of text matching (usually regex stuff) and implement various syntax highlighting (etc...) off of that. It is not suitable for something as complex and context-aware as the 4GL. Useful links:

- XText
- XText
- Antlr4 to xtext
- rewrite ANTLR .g grammar in XText

The idea is that it would not be too hard to do this if your grammar is very simple and has few actions and little to no semantic/syntactic predicates. FYI, progress.g is hugely dependent upon actions and semantic predicates so this means it would probably be infeasible to reuse the existing grammar.

The original article at the top of this node suggests looking at Geppetto which is an Eclipse plugin that uses an external JRuby compiler to implement a Puppet development environment in Eclipse. It is implemented in a way that the backing tools are generic and can be used from other IDEs or tooling environments. This is the same requirement that we have.

I think we will need to implement a plugin using the "platform" itself. This is the underlying API used by any plugin, including those which provide a perspective. We will also need to integrate with the Java Development Tools (JDT is one of the main sub-projects of Eclipse; it provides the Java perspective).

Eclipse Documentation (click into a specific version and you will want to look for "Platform Plug-in Developer Guide", "JDT Plug-in Developer Guide" and "Plug-in Development Environment Guide")

Eclipse Main Project Wiki Page (pretty high level, but it explains the core components)

Java Development Tools (this is a nice article with some details about the JDT subproject)

JDT Github Projects (for example, the core JDT support or JDT UI or JDT debugger)

#3 - 08/15/2021 11:07 AM - Greg Shah

Questions/Items to Investigate

- Generic Editor
- Language Server Protocol (LSP)
  - The Language Server Protocol (LSP) is a platform independent protocol for creating language support that can work across various IDEs (VSCode, Eclipse IDE, Eclipse Che, vim...).
  - It is an open standard, originally created by Microsoft for their internal use and then made more generic through work with Eclipse, IBM and Red Hat. Eclipse now supports it quite well and it can be easily plugged in to the Generic Editor support to provide a pretty rich environment.
  - Eclipse has built-in support for integrating with the LSP. It also has a kind of library to help implement support for a language using the LSP, called LSP4E (see the LSP For Eclipse Github Project.
  - You still need to create a plugin, but it can be quite small (Using Language Servers to Edit Code in the Eclipse IDE).
  - To the degree that there are limitations, some articles suggest that some extension points and traditional Eclipse plugin support can be added to extend the "generic" LSP support and make the result complete.
  - **What are the limitations?**
    - Eclipse support in Xtext as LSP + Generic Editor? suggests there is a trade off between light/universal vs complete rich edting support.
    - XText LSP Support
    - Some articles suggest that syntax highlighting is not yet supported. The recommendation is that TextMate can be used for that (TextMate for Eclipse Github project).
  - Why You Should Know the Language Server Protocol
- Debug Adapter Protocol
  - The Debug Adapter Protocol is a platform independent protocol for separating the debugging back-end from the debugger UI. As such it allows the same back-end debugging support to be used by multiple IDEs.
  - It is an open standard, originally created by Microsoft for the VSCode product. It is being adopted by multiple IDEs now and it is a kind of companion to the Language Server Protocol.
  - Adopting the Debug Adapter Protocol in Eclipse IDE
    lipse-platform-generic-editor
- Eclipse Che
  - This is a web-based IDE.
  - I am wondering if this is the more strategic option over the long term.
  - It is definitely the less mature option at this point. And it is less well known.
- Eclipse Theia
  - Open source, extensible IDE.
  - Builds upon VSCode but is supposedly more modular/extensible.
  - Supports VSCode extensions.
  - Can run a web-based IDE or desktop IDE from the same source implementation. To deliver on the desktop, you would use something like Electron.
  - Leverages both LSP and DAP.
  - Used in Eclipse Che (starting in v7).
  - Eclipse Theia
  - Eclipse Theia is the Next Generation Eclipse Platform for IDEs and Tools!
  - The Eclipse Theia IDE vs. VS Code (FYI, Theia is based on the VS Code open-source project and then extends/expands from there)

#4 - 09/08/2021 12:28 PM - Greg Shah

From Hynek:

> Btw. the debugging experience in Developer Studio is pretty lousy. For example I couldn't add new breakpoints doubleclicking in the editor or using the menu, I had to add it manually in the debuggers window (enter the full procedure file path and line number). Also inspecting variable values by hover in the editor doesn't work either, I had add the expression manually in the watch window.

#5 - 09/09/2021 01:10 PM - Greg Shah

> I couldn't add new breakpoints doubleclicking in the editor or using the menu, I had to add it manually in the debuggers window (enter the full procedure file path and line number).

I was pretty surprised at this limitation. Normally, this is a very simple integration that is supported for plugins that have a source code editor view. I looked in the Developer Studio documentation and it seems it should in fact support the normal breakpoint capabilities (e.g. adding via a double click in the source editor). I think the reason it is not working for you is probably that there is no project created for the source you are debugging. Without a project, the source editor integration won't work.

#6 - 05/04/2022 10:47 AM - Greg Shah

- Related to Feature #6319: IntelliJ plugin added

#7 - 01/26/2023 02:02 PM - Greg Shah

In April 2022, the official word is that Eclipse Theia is the next generation of Eclipse!. This doesn't mean that the Eclipse desktop version is going away, but rather that Theia is more strategic.

The more I dig into this, the following seems pretty clear to me:

1. Writing a language server (i.e. using the LSP and DAP) is the best long term approach to implementing common code for multiple IDEs. VS Code and Theia both use this "natively" as the back-end. Eclipse desktop and IntelliJ both have plugins can allow LSP to be used. Full LSP support in Eclipse desktop and IntelliJ might require us to write a more complex plugin. In other words, there may be more or less work in the "client side" (a.k.a. a plugin) of a LSP depending on the level of LSP support that exists for a platform.
2. We will need to decide if the maturity of the Eclipse JDT platform justifies writing an SWT-based plugin at the time when the world is already shifting to the web. Considering that we already have a web UI written for the report server AND we are very interested in expanding it, the web is very attractive. It could be a common implementation for all of this. In addition we've considered the value of a web-based IDE for implementing conversion projects. The idea here is that we could create a portal that would make it very easy for people to try out FWD. A purely web based approach would easily enable this idea. I'm leaning toward using the web based approach rather than investing in SWT.

#8 - 02/10/2023 09:08 AM - Greg Shah

- Related to Feature #1757: update ANTLR to latest version added

#9 - 02/10/2023 09:34 AM - Greg Shah

See #1757-9 for some analysis of how auto-completion may require a move to ANTLRv4.