History

History

Table of Contents

History

Vala Programming/Header

History[edit]

Language Features[edit]

The syntax of Vala is similar to C#, modified to better fit the GObject type system. Vala supports modern language features as the following:

- Interfaces
- Properties
- Signals
- Foreach
- Lambda expressions
- Type inference for local variables
- Generics
- Non-null types
- Assisted memory management
- Exception handling
- Type modules (Plugins)

Vala is designed to allow access to existing C libraries, especially GObject-based libraries, without the need for runtime bindings. All that is needed to use a library with Vala is an API file, containing the class and method declarations in Vala syntax. Vala currently comes with experimental bindings for GLib and GTK+. It's planned to provide generated bindings for the full GNOME Platform at a later stage.

Using classes and methods written in Vala from an application written in C is not difficult. The Vala library only has to install the generated header files and C applications may then access the GObject-based API of the Vala library as usual. It should also be easily possible to write a bindings generator for access to Vala libraries from applications written in e.g. C# as the Vala parser is written as a library, so that all compile-time information is available when generating a binding.

Why Vala?[edit]

Many developers want to write GNOME applications and libraries in high-level programming languages but can't or don't want to use C# or Java or Python or Ruby or Perl or Lua or Haskell or Erlang or C++.... for various reasons, so they are stuck with C without syntax support for the GObject type system. The Vala compiler allows developers to write complex object-oriented code rapidly while maintaining a standard C API and ABI and keeping the memory requirements low.

C# and Java libraries can't be used the same way as native GObject libraries from C and other languages and can't be accepted as part of the GNOME Platform. Managed applications also suffer from usually higher memory requirements which is not acceptable in some situations.

The Vala compiler (valac) produces C source and header files from Vala source files as if you had written your library or application directly in C. Using a Vala library from a C application won't look different than using any other GObject-based library. There won't be a vala runtime library and applications can distribute the generated C code with their tarballs, so there are no additional run- or build-time dependencies for users.

Caveats[edit]