Development Kit

Page History

DevKit Overview

The DevKit is a toolkit that makes it easy to build and use native C/C++ extensions such as RDiscount and RedCloth for Ruby on Windows.

Simply download, double-click, choose an installation directory, run the Ruby install helper script, possibly tweak a `config.yml` file for your system specifics, and you’re ready to start using native Ruby extensions.

DevKit Background

One of the challenges many Ruby on Windows users have is how to easily use native RubyGems from the community such as the rdiscount or curb gems. In many cases this isn’t a problem because the extension author has provided a binary gem compatible with the users Ruby environment. In other cases, only the extension source code is available in the gem and the user is expected to have a “sane” environment installed and configured in order to build and use the native gem. A “sane” environment typically means a system with `make, gcc, sh` and similar *nix build tools installed.

This expectation of a “sane” build environment is not always true for Windows users and is the primary reason why the DevKit was created. The DevKit, based upon MSYS and MinGW components, is an easy-to-install and easy-to-use solution for quickly setting up a “sane” build environment, enabling you to use most native RubyGems developed by the Ruby community.

We use the DevKit internally for the RubyInstaller project. The DevKit is used to build MRI Ruby and the required dependencies needed to deliver the RubyInstaller for Windows. Many community members also use the DevKit as their “foundation” toolkit for building other native software.

The DevKit is currently available for download as a self-extracting archive and a Windows installer is in development.

Requirements

- Window XP or greater (32 or 64-bit).
- A Ruby installation from RubyInstaller. This is important as it will not work with any other version of Ruby (e.g. old One-Click)
- ***[optional]*** For RubyInstaller’s previous to `1.9.2-p0` you may feel more comfortable using the `DevKit-3.4.5-CCYYMMDD-sfx.exe` as the earlier RubyInstallers were built using MSys/MinGW v3.4.5. Currently no issues have been reported when either the 4.5.0 or 4.5.1 DevKit’s are used with previous RubyInstaller versions.

Installation Instructions

While installation is (in general) simple, please ensure you carefully follow each step below.

1. PREPARATION

- If you previously installed the legacy DevKit `devkit-3.4.5r3-20091110.7z`, its artifacts were extracted into each Ruby installation and need to be manually removed. Remove the `gcc.bat, make.bat, and sh.bat` stub batch files in `<RUBY_INSTALL_DIR>\bin` and the `<RUBY_INSTALL_DIR>\devkit` subdirectory for each Ruby installation using the legacy DevKit.
- If you previously installed one of the legacy self-extracting DevKit’s, follow the SFX DevKit upgrade instructions.

2. DOWNLOAD FILES

- The current DevKit is available at the RubyInstaller download page with older versions available at the archives page. As backup, check our GitHub downloads page.

3. EXTRACT FILES

- Left double-click the self-extracting executable (SFX) downloaded from Step 2 and choose a directory (without spaces) to install the DevKit artifacts into. For example, `C:\DevKit`. NOTE: the SFX is really a 7-Zip archive with a bit of embedded magic. If you already have 7-Zip installed, you can simply right-click it and extract it's contents as you would a normal 7z archive. In the instructions that follow, the directory that you selected is identified as `<DEVKIT_INSTALL_DIR>`.

4. RUN INSTALLATION SCRIPTS

- `cd <DEVKIT_INSTALL_DIR>` from Step 3 above.
- `ruby dk.rb init` to generate the `config.yml` file to be used later in this Step. Your installed Rubies will be listed there (only those installed by a RubyInstaller package are detected at present).
- edit the generated `config.yml` file to include installed Rubies not automagically discovered or remove Rubies you do not want to use the DevKit with.
- ***[optional]*** `ruby dk.rb review` to review the list of Rubies to be enhanced to use the DevKit and verify the changes you made to it are correct.
- finally, `ruby dk.rb install` to DevKit enhance your installed Rubies. This step installs (or updates) an `operating_system.rb` file into the relevant directory needed to implement a RubyGems `pre_install` hook and a `devkit.rb` helper library file into `<RUBY_INSTALL_DIR>\lib\ruby\site_ruby`. NOTE: you may need to use the `--force` option to update (with backup of the originals) the above mentioned files as discussed at the SFX DevKit upgrade FAQ entry.

5. TEST INSTALLATION

- Confirm your Ruby environment is correctly using the DevKit by running `gem install rdiscount --platform=ruby`. RDiscount should install correctly and you should see `Temporarily enhancing PATH to include DevKit...` in the screen messages. Next run `ruby -rubygems -e "require 'rdiscount'; puts RDiscount.new('**Hello RubyInstaller**').to_html"` to confirm that the rdiscount gem is working.

TROUBLESHOOTING

Please refer to Troubleshooting for known issues or RubyInstaller Google Group for further help.

Example Native RubyGem Installations using the DevKit

Once you've installed the DevKit you'll discover there are but a few different types of native installation scenarios to deal with.

- The Typical Install Scenario – most of the native RubyGems you're likely to encounter can be installed by running something similar to `gem install bson_ext --platform=ruby`. It's crucial that you include the `--platform=ruby` option to force RubyGems to build the native gem rather than potentially installing an incorrect binary gem.

- The Manual Install Scenario – sometimes a native RubyGem requires additional artifacts to already exist on your system and be usable. One such gem is curb which requires curl's native header and library files in order be built and installed. In this scenario you need to tell RubyGems where to find those headers and library files by using a command similar to `gem install curb --platform=ruby -- --with-curl-lib="C:/curl/bin" --with-curl-include="C:/curl/include"`. In addition, the curl library found in the directory specified with `--with-curl-lib` must be on your `PATH` for use at runtime by the curb gem.

- The Hacky Developer Scenario – a developer building native gems wants to be able to quickly test that their `extconf.rb` file used to create a `Makefile` for the native library works correctly. To shorten the development cycle, the DevKit enables the developer to run `ruby -rdevkit extconf.rb`.

Accessing the Full MSys/MinGW Environment

To access all of the MSYS/MinGW tools from a Command Prompt shell, run the `devkitvars.bat` helper script (lives in `<DEVKIT_INSTALL_DIR>`) to update the shell's `PATH` environment variable. Similarly, if you're using PowerShell, run the `devkitvars.ps1` helper script.

Building the DevKit

As the DevKit is a fundamental part of the RubyInstaller build recipes, you can always build your own DevKit by simply:

1. `git clone git://github.com/oneclick/rubyinstaller.git`
2. `cd rubyinstaller`
3. `rake devkit sfx=1`

Next Steps

Now that you've installed the DevKit, you should be able to use many of the native C RubyGems without problems.

However, some extensions may require additional header and library files to also be installed, and still other extensions may need additional work done by the authors in order to get them to compile with the MSys/MinGW based DevKit. If you have problems, please drop us a line at our RubyInstaller Google Group

Last edited by luislavena, June 05, 2011

GitHub

About
Blog
Features
Contact & Support
Training
Site Status

Tools

GitHub for Mac
Issues for iPhone
Gist: Code Snippets
Enterprise Install
Job Board

Extras

GitHub Shop
The Octodex

Documentation

GitHub Help
Developer API
GitHub Flavored Markdown
GitHub Pages

Terms of Service Privacy Security

Powered by the Dedicated Servers and Cloud Computing of Rackspace Hosting®