Over the past few years, Rust has proven to be an incredibly stable and accessible option for those wishing to write native extensions for Ruby. Rust is particularly well-suited for Ruby for a few reasons:

1. Integration is straightforward, since it is easy to expose a C compatible API using extern C
2. Rust is a bit more high-level, and approachable for Ruby developers since it prevents many of the classic C footguns (use-after-free, etc)
3. The landing page can explain the benefits better than me. Check out [https://www.rust-lang.org/](https://www.rust-lang.org/) if you aren't fully convinced.

One of the biggest pain points of creating a Gem with a Rust extension is integrating with Gem::Ext::Builder. There are currently a couple of solutions, and they rely on the RakeBuilder interface. I have personally used [https://github.com/malept/thermite](https://github.com/malept/thermite), which is a very good solution, but still requires external dependency and does not integrate as well as native builders at the moment.

Currently, Gem::Ext::Builder supports four builder types: ExtConfBuilder, ConfigureBuilder, RakeBuilder, and CmakeBuilder. I propose we add a new builder, CargoBuilder, which will detect a cargo.toml file and build the gem from that. This ease the burden of developing and publishing Rust extensions for Ruby users.

**Potential Problems**

1. If the user does not have cargo installed, building the extension will not work. This is an issue, but is also an issue for the CmakeBuilder. One way to mitigate this issue would be to make it easier to build static binaries for Rust extensions.

2. The lack of a standard Ruby interface for Rust extensions might cause pain for developers. Currently, many folks use [https://github.com/steveklabnik/ruby-sys](https://github.com/steveklabnik/ruby-sys) to interface with the C Ruby API. It might make sense to make this "officially" supported at some point, but I don't think it has to be done immediately.

Please let me know your thoughts. I would be very excited to see Ruby take this step. I don't think I know enough about the Ruby extension building process or the Rust compilation process to actually implement this, so hopefully we could find a volunteer to step up. Thanks for reading :)

---

**Notable Gems using Rust**

1. [https://github.com/danielpclark/faster_path](https://github.com/danielpclark/faster_path)
2. [https://github.com/ianks/fast_woothee](https://github.com/ianks/fast_woothee) -- shameless plug
3. [https://github.com/tildeio/helix](https://github.com/tildeio/helix) -- a framework for building Rust extensions
4. [https://github.com/danielpclark/rute](https://github.com/danielpclark/rute) -- ruby/rust glue
5. [https://github.com/steveklabnik/ruby-sys](https://github.com/steveklabnik/ruby-sys) -- low level bindings for ruby
6. [https://github.com/malept/rusty_blank](https://github.com/malept/rusty_blank) -- String#blank? with rust

---

**#2 - 04/10/2019 05:32 AM - shevegen (Robert A. Heller)**

I can not speak for the ruby core team; it may be best to actually ask matz directly, but let's ignore this for the moment. I think, viewed objectively, there would be no reason to avoid/ignore Rust. Ruby's primary internal focus is on C though; even C++ may be a second-class citizen compared to C.

I am not sure whether it is feasible to use an existing project (say, ruby-sys) and make it anything official per se, because that depends on who is going to maintain the code, in particular when there are changes in Rust that may force changes in the API from the ruby side too. Again, I do not know, but I think that this may be a possible problem. Another slight issue may be that not that many from among the
core team actually know and use Rust; probably only a small portion do so. (It may also have to compete with other time investments.)

I don't know if maintaining the code for getting rust-support is a huge issue but what if bindings may become outdated? This actually happened when qt+kde switched from version three, to four and then to five. Most ruby-qt stuff is still at qt/kde version 4, and the transition into 5 killed this mostly completely (although it was the initial change to qt4 that did most damage); ruby-gtk bindings on the other hand were diligently updated slowly, and as a consequence are a lot more useful these days (my personal opinion here).

Anyway, I don't think anyone may be against the described goal, just the way and what has to be done is not completely clear (to me at the least; I guess we may have to find someone who knows both Rust and Ruby very well).

### #3 - 04/10/2019 05:35 AM - duerst (Martin Dürst)
- Status changed from Open to Third Party's Issue

Ruby includes rubygems, but this is imported, see e.g. [https://svn.ruby-lang.org/cgi-bin/viewvc.cgi?view=revision&revision=67415](https://svn.ruby-lang.org/cgi-bin/viewvc.cgi?view=revision&revision=67415). So improvements to rubygems should be proposed and discussed upstreams.