Ruby master - Misc #13072

Current state of date standard library

12/25/2016 01:04 PM - zverok (Victor Shepelev)

Status:	Open	
Priority:	Normal	
Assignee:		

Description

The facts that I've been able to gather (not supported by links, so please forgive me if I am misquoting/misunderstanding):

- date library was initially developed and maintained by Tadayoshi Funaba, who was the "single point of truth" for its design and features;
- for at least year, initial creator/maintainer of the library is inactive in Ruby community, so library mostly considered unmaintained;
- as far as I can "sense"/guess from ticket responses about the library, the core team doesn't see it as crucial/important to maintain.

At the same time, the library provides:

- Widely and extensively used Date class;
- Pretty controversial DateTime class, which has a huge feature intersection but almost no compatibility with core Time class;
- Date parsing functionality (Date._parse), which is also used by lib/time.

The latter also leads to a really confusing situation, where one of the core Ruby classes has "optional additional functionality" in stdlib.

Overall, the situation looks pretty "dirty" (as in "dirty code"), and seems like needing improvement.

WDYT about this plan (for Ruby 2.5, for ex.):

- make Date and Date._parse parts of language core (with probably renaming _parse to something more readable, or even extracting something like Date::Parser module);
- merge DateTime and Time (while preferring Times interface where possible);
- on the way, gather all requests/bugs from this tracker, related to dates and times parsing, representing and so on.

Related issues:

Related to Ruby master - Feature #5481: Gemifying Ruby standard library	**Closed**
Related to Ruby master - Feature #13183: Gemify date	**Closed**
Related to Ruby master - Misc #14274: Merge Std-Lib Time Class into Core	**Open**

History

#1 - 12/26/2016 12:41 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #5481: Gemifying Ruby standard library added

#2 - 01/20/2017 10:36 AM - naruse (Yui NARUSE)

Victor Shepelev wrote:

> The facts that I've been able to gather (not supported by links, so please forgive me if I am misquoting/misunderstanding):
>
> - date library was initially developed and maintained by Tadayoshi Funaba, who was the "single point of truth" for its design and features;
> - for at least year, initial creator/maintainer of the library is inactive in Ruby community, so library mostly considered unmaintained;
> - as far as I can "sense"/guess from ticket responses about the library, the core team doesn't see it as crucial/important to maintain.
>
> At the same time, the library provides:
>
> - Widely and extensively used Date class;
> - Pretty controversial DateTime class, which has a huge feature intersection but almost no compatibility with core Time class;
> - Date parsing functionality (Date._parse), which is also used by lib/time.
>
> The latter also leads to a really confusing situation, where one of the core Ruby classes has "optional additional functionality" in stdlib.
>
> Overall, the situation looks pretty "dirty" (as in "dirty code"), and seems like needing improvement.

I agree with this understandings.
And some people continually discussing about date library.

> WDYT about this plan (for Ruby 2.5, for ex.):
>
> - make Date and Date._parse parts of language core (with probably renaming _parse to something more readable, or even extracting something like Date::Parser module);
> - merge DateTime and Time (while preferring Times interface where possible);
> - on the way, gather all requests/bugs from this tracker, related to dates and times parsing, representing and so on.

Current my thoughts is:

- Time is kept in core
- date library will be separeted from Ruby repository (stdlib) to date.gem.
- Time.parse is kept even though the API is too ugly and heuristic (use Time.iso8601 and so on)
  - some of Time.parse's implementations are in ext/date/date_parse.c. It is needed to be separated into ext/time or something.
  - I haven't consider about Date._parse yet. Maybe it needs something to be care.

#3 - 01/20/2017 11:16 AM - zverok (Victor Shepelev)

> Time is kept in core
> date library will be separeted from Ruby repository (stdlib) to date.gem.

What problems will it solve? Just date becoming "not my problem" for core/stdlib maintainers?
Date is widely used, and its compatibility with Time (for comparisons, for example) is a huge pain. Being separated into the gem, Date will become incompatible forever (because separate gem definitely can't introduce changes in Time#< and similar methods). The situation will be even worse than now.

#4 - 01/25/2017 05:45 AM - naruse (Yui NARUSE)

Victor Shepelev wrote:

> > Time is kept in core
> > date library will be separeted from Ruby repository (stdlib) to date.gem.
>
> What problems will it solve? Just date becoming "not my problem" for core/stdlib maintainers?
> Date is widely used, and its compatibility with Time (for comparisons, for example) is a huge pain. Being separated into the gem, Date will become incompatible forever (because separate gem definitely can't introduce changes in Time#< and similar methods). The situation will be even worse than now.

Date and Time are incompatible both its philosophy, implementation, and API.
If we make Date compatible with Time, it needs Date to introduce incompatible API change.
If so, Date users should just migrate to Time.
Time has enough feature to be migrated from Date. (if not, such feature must be added into Time)

There seems still exist an issue, a class named "Date" for the class which represents date type of RDB.
But I want to discuss about that without current ext/date library.

#5 - 02/02/2017 08:37 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13183: Gemify date added

#6 - 02/11/2017 09:05 AM - shevegen (Robert A. Heiler)

Just wanted to add - for a new user of ruby, it may be confusing to
know the difference between Time, Date, DateTime and DuckTime (ok
the last one is made up but you get the point).

So from that point of view, it may be easier to unify on something,
no matter what, and encourage people to use it (or, if you want
to make it a cleaner break, integrate functionality into one
namespace and discourage the other namespaces, but I have no
preference either way, just pointing out the proliferation of
different names there).

#7 - 02/18/2017 07:05 PM - stomar (Marcus Stollsteimer)

> date library will be separeted from Ruby repository (stdlib) to date.gem

I think the concept of date (without time) is so important that it should not simply be unbundled into a gem.

I'm concerned about the increasing tendency to unbundle core functionality, and fear that Ruby will loose some of its usability and attractiveness as an all-purpose language, especially for simple, easy to distribute, single-file scripts for e.g. administrative tasks, text processing, or data analysis.

Please keep in mind that installing gems is not always that easy, for instance without root access or on non-Unix-like operating systems -- even more so for native gems. [Please forgive the (partial) double posting with #13221.]

The current situation with Time, Date, DateTime is indeed confusing, but instead of removing date offhandedly please try to find a solution where the core functionality is kept somewhere in the stdlib (not necessarily where it is now).

#8 - 04/05/2017 03:44 AM - shyouhei (Shyouhei Urabe)

stomar (Marcus Stollsteimer) wrote:

> > date library will be separeted from Ruby repository (stdlib) to date.gem
>
> I think the concept of date (without time) is so important that it should not simply be unbundled into a gem.

Separating repository does not directly mean unbundling.

For instance we recently separated our openssl library to its dedicated repository. But this doesn't mean we unbundled it. Because rubygems depends that library, we cannot but ship them altogether.

Quite generally speaking, there are currently 4 kinds of situation for a formerly-standard library:

- Ones that are still remaining as they were before.
- So-called "default gems": ones that resides as source codes in the shipped tarball, with gemspec.
- So-called "bundled gems": ones that resides as .gem file in the shipped tarball. They don't need network access to install.
- Ones that are not bundled any longer. Installing them needs network access.

All but the last one are currently considered somewhat "standard". And I don't think Date would go straight to the last kind.

> I'm concerned about the increasing tendency to unbundle core functionality, and fear that Ruby will loose some of its usability and attractiveness as an all-purpose language, especially for simple, easy to distribute, single-file scripts for e.g. administrative tasks, text processing, or data analysis.
>
> Please keep in mind that installing gems is not always that easy, for instance without root access or on non-Unix-like operating systems -- even more so for native gems. [Please forgive the (partial) double posting with #13221.]

This concern is valid.

But what is discussed in this thread is to transplant some part of ext/date functionality to Time, to make them separate. Once that happened, I think there are chances for Date to be a pure-ruby library again (note: it once was).

> The current situation with Time, Date, DateTime is indeed confusing, but instead of removing date offhandedly please try to find a solution where the core functionality is kept somewhere in the stdlib (not necessarily where it is now).

Creating a gem and to unbundle it is a separate thing, and gem itself should have no side effect I believe. We can arrange the way how Date library would be placed in a future.

#9 - 04/09/2017 06:59 PM - stomar (Marcus Stollsteimer)

shyouhei (Shyouhei Urabe) wrote:

> All but the last one are currently considered somewhat "standard". And I don't think Date would go straight to the last kind.

Thanks for your clarifications, I guess I got a wrong impression. (I believe I read the ticket for removing curses shortly before, which seemed to have been kind of sudden.)

#10 - 01/05/2018 09:21 PM - naruse (Yui NARUSE)

- Related to Misc #14274: Merge Std-Lib Time Class into Core added