Rubygems

Lucas Nussbaum
Rubygems

The standard to install Ruby libraries

Features :

- no intermediate packagers (e.g Linux distro)
  => no delay
  => always the latest version (even git snapshots)
- cross-platform, works the same on Windows, Mac OS X, Linux

Other languages have similar tools :

- Perl -> CPAN
- Python -> eggs

Fabulous tool for developers !
Rubygems: what about users?

users = end users, sysadmins, etc.

Would like: **standard, common way to install software**

- ease of use
- integrates well with infrastructure (deployment, local mirrors, etc)
- easy security updates

On Linux: provided by their **distribution’s package manager**
They provide distribution methods for both use cases:

- **Perl**:
  ```
  perl Makefile.pl && make && make install
  ```
- **Python**: distutils

**Ruby** imposes the use of **rubygems** to everyone, both developers and users!
"But you don’t need to use gems!" WRONG!

See yourself:
Try to install rails applications without rubygems
In many libraries, needed changes:

@@ -5,7 +5,7 @@

-require "rubygems"
+# require "rubygems"
 require "highline/import"
Other problems (mostly consequences)

Those are technical problems, not social problems

- Rubygems ignores non-rubygems libraries when trying to satisfy dependencies
  (Necessary: doesn’t know the version of the lib)
- Rubygems encourages strict dependencies on a specific version
  (API versioning != library versioning)
  - you end up with many versions installed
- Rubygems doesn’t encourage stable APIs
  - users encouraged/forced to install the latest version
- Standardization on some external websites
  - Rubyforge, Github
Conclusion

Solution: vendorization? "let’s just put everything in vendor/!
WRONG

Rubygems and Git are very good developer tools
  But they are not good generic tools for distribution of ruby libraries

The library problem has been solved everywhere years ago.
  Just need to create ruby equivalents to those solutions
  some tools already exist: setup.rb