Fedora 17 and Ruby

Bohuslav „Slávek“ Kabrda
Thanks to all members of Fedora Ruby SIG: Vít Ondruch, Mamoru Tasaka, Mo Morsi and many others.

Fedora 17 uses MRI Ruby 1.9.3 (Fedora 16 has Ruby 1.8.7):
• Some backward incompatible changes in the language itself and in the Ruby abi, too.
• All of the packages (not only Gems) using Ruby were rebuilt – cca 300 packages.

Filesystem layout changes:
• Conformance with FHS.
  ◦ Binary libraries and extensions of Gems are placed under /usr/lib (resp. /usr/lib64)
  ◦ Pure Ruby libraries and Gems are placed under /usr/share.
• Three places for Gems - /usr, /usr/local, ~/.gem (see below section for rules about these directories).
• RubyGems library now under /usr/share/rubygems.

Using Gems:
• Three methods to install, three different places for Gems (RubyGems were modified to install to the mentioned directories and load Gems from them, everything works out of the box):

<table>
<thead>
<tr>
<th>How to install</th>
<th>Place</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ gem install foo</td>
<td>~/.gem</td>
<td>Only the user can see and use these Gems.</td>
</tr>
<tr>
<td>$ bundle install</td>
<td></td>
<td></td>
</tr>
<tr>
<td># gem install foo</td>
<td>/usr/local</td>
<td>System wide Gems, all users can see and use them.</td>
</tr>
<tr>
<td># bundle install</td>
<td></td>
<td></td>
</tr>
<tr>
<td># yum install rubygem-foo</td>
<td>/usr</td>
<td>System wide Gems supported by Fedora, the gem command can't touch them by default, only RPM can.</td>
</tr>
</tbody>
</table>

• Pure Ruby code placed under /usr/share/gems and /usr/local/share/gems, extensions (if any) placed under /usr/lib[64]/gems and /usr/local/lib[64]/gems (Gems under home are installed into a single directory, as there is no need to split them to satisfy FHS).

Changes from packager's perspective:
• updated packaging guidelines (https://fedoraproject.org/wiki/Packaging:Ruby)
• new macros for both non-Gem libraries and Gems (see the guidelines for reference)

Suggested usage:
• Use user-installed Gems for development.
• Use RPM Gems for stabilization of the application and production deployment (good support, backported security patches, stable versions).

Next steps:
• Deeper integration with bundler (currently installs newest versions in Gems, even if installed Gems already satisfy dependencies – bad for RPM Gems, as they might get substituted by newer (non-RPM) versions).
• Integration of system RubyGems under /usr/share/rubygems with JRuby – deduplication, better maintainability, using pure Ruby Gems for both implementations.
• Packaging Rubinius(?)