Week 5 Lecture 15

Business

Helpful Resources

- http://gembundler.com/
- http://rubygems.org/

What's in this lecture?

- Gems
- RubyGems
- Bundler

The story so far...

- Code is organized into classes
- MVC design pattern
- Libraries are encapsulated ‘chunks’ of functionality

Enter Gems

- Packaged Ruby libraries (or applications)
- Three components:
  - name
  - version number
  - code
- Most gems have dependencies on other gems, or system packages

Gems: Name

- Unique
- Contain no spaces
- Set by the author
- Used as identifier for search and installation

Gems: Version Number

- X.Y.Z where:
  - X -- major version release. May contain breaking changes
  - Y -- minor version release. May contain new, non-breaking functionality
  - Z -- patch level release. Bug fixes only.

Unfortunately

- That doesn’t always happen in practice (see: rake)

Gems: The Code

- Everything (important) is stored in the /lib directory of the gem
- You can inspect the code of a gem by:

```
$ bundle show gem_name
```

- and opening the gem's files from that path in your text editor

Examples of Gems

- Authentication
- XML Parsing
- Rails
- ORM
- client-side validations

Distribution

- RubyGems is Ruby's package manager
- System for publishing and distributing gems
- What does that mean?
  - Author writes gem
  - Tells RubyGems gem name, version number, and code
  - RubyGems then makes it available for search and download by other Ruby developers

The Source Index

- RubyGems keeps track of all of this with a source index
- When you need to search for or install, you are looking through the nearest source index
- Typically you can only do this with an internet connection -- offline is possible though

Bundler

- Understanding the problem:
  - multiple gem versions causes headaches
  - an application tries to load a gem, and accidentally grabs the wrong version!
  - code breaks!   x_x
- Team needs to work off same gemset

Bundler

- Need a way to:
  - manage all of the gems for an application
  - ensure that the team uses the same gems
  - provide an easy way to lock versions
  - resolve dependencies (even conflicting)

Bundler: Gemfile

- Gemfile
  - lives at root directory of the application
  - specifies gems to be used
  - has options for
    - version number
    - source specification

Bundler: Gemfile.lock

- When you run ‘bundle install’, bundler:
  - creates a mapping of every specified gem, their version, and the names/versions of their dependencies
  - downloads gems from RubyGems
  - creates a file ‘Gemfile.lock’ which specifies the app gems and their versions

New Gems

- If you add or subtract from Gemfile
  - Run bundle install
  - commit the new Gemfile and Gemfile.lock

Updating Gems

- ‘bundle update’ without arguments tries to update every gem to their latest version (potentially dangerous!)
- ‘bundle update rails --version=3.0.7’ will update _just_ the gem ‘rails’ (and rebuild its dependency tree)

When starting a new project...

- Create an .rvmrc file that specifies the ruby version and the gemset
- install bundler
- create Gemfile with the required gems
- run bundle install
- commit Gemfile.lock

Exercises

- http://railscasts.com/episodes/201-bundler
- http://railscasts.com/episodes/245-new-gem-with-bundler
- Create a basic rails app that uses the gem devise