Ruby Monstas

Session 18

Agenda

Recap
Rubygems
Bundler
Exercises

Recap

Enter: ERB!

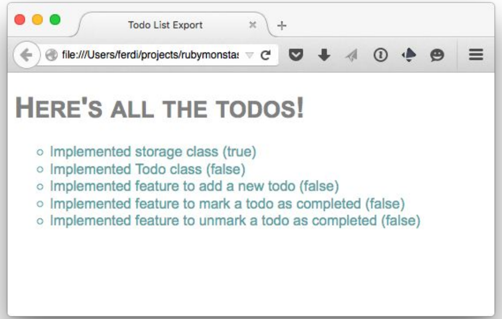
todos.html.erb

```
<!DOCTYPE html>
<html>
  <head>
    <title>Todo List Export</title>
  </head>

  <body>
    <h1>Here's all the todos!</h1>

    <ul>
      <% @todos.each do |todo| %>
        <li>
          <%= todo.title %> (<%= todo.completed? %>)
        </li>
      <% end %>
    </ul>
  </body>
</html>
```

erb_exporter.rb

```
require 'erb'

class ErbExporter

  def initialize(todos)
    @todos = todos
  end

  def export(file_name)
    File.open(file_name, 'wb') do |output_file|
      template_source = File.new('todos.html.erb').read
      erb = ERB.new(template_source)
      rendered_html = erb.result(binding)
      output_file.write(rendered_html)
    end

    puts "Written HTML export to file #{file_name}."
  end

end
```

Simple CSS Rules



HERE'S ALL THE TODOS!

- Implemented storage class (true)
- Implemented Todo class (false)
- Implemented feature to add a new todo (false)
- Implemented feature to mark a todo as completed (false)
- Implemented feature to unmark a todo as completed (false)

export.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Todo List Export</title>
    <link rel="stylesheet" type="text/css" href="export.css" />
  </head>

  <body>
    <h1>Here's all the todos!</h1>

    <ul>

        <li>
          Implemented storage class (true)
        </li>

        <li>
          Implemented Todo class (false)
        </li>

        <li>
          Implemented feature to add a new todo (false)
        </li>

        <li>
          Implemented feature to mark a todo as completed (false)
        </li>

        <li>
          Implemented feature to unmark a todo as completed (false)
        </li>

    </ul>
  </body>
</html>
```

export.css

```
body {
  font-family: "Arial";
}

h1 {
  font-variant: small-caps;
  color: grey;
}

ul {
  list-style-type: circle;
}

li {
  color: CadetBlue;
}
```

Rubygems

The Ruby package manager

Package manager

The standard library is not enough!

What do?
Package manager!

The Ruby package manager: “Rubygems”
The packages: “gems”

Basic usage

Rubygems comes with Ruby. It’s already installed on your system.

```
gem install <gem-name>
gem list
gem uninstall <gem-name>
```

Resources

http://rubygems.org/

http://bestgems.org/

https://www.ruby-toolbox.com/

The problems with Rubygems…

What if I use 100 gems in my project?

What if I want version 3.1.8 of the `haml` gem and not the current (4.0.7)?

My colleague installed version X of the gem and I'm using version Y!

Bundler

Ruby dependency manager

Enter bundler!

`gem install bundler`

Edit `Gemfile`

`bundle install` (or just `bundle`)

Gemfile

```ruby
source 'https://rubygems.org'

gem 'pg'                          # any version of the pg gem

gem 'rails', '4.2.4'              # exactly version 4.2.4 of the rails gem

gem 'sass-rails', '>= 5.0'        # any version higher or equal to 5.0 of sass-rails

gem 'uglifier', '< 2.7.2'         # any version before 2.7.2 of uglifier

gem 'draper', '~> 2.1.0'          # any version of draper higher or equal to 2.1.0, but
                                  # smaller than 2.2.
```

Semantic versioning

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards-compatible manner, and
3. PATCH version when you make backwards-compatible bug fixes.

bundle install

1. looks at Gemfile
2. checks for available gem versions on rubygems.org
3. analyses and resolves the versions of gems
4. installs gems
5. generates the Gemfile.lock file

Time to practice