Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Chapters

How can I internationalize my application? {#i18n} ¶ ↑

We will rely on the `i18n` gem to handle internationalisation of strings and objects, and to manage fallbacks on available locales

```
require 'i18n'
require 'i18n/backend/fallbacks'
```

The following configuration is necessary on I18n so that:

- it can fallback on other locales if the requested one is not available (ie: translation does not exist).
- all the transalations are read from YAML files located in the `locales` directory

```
configure
  I18n::Backend::Simple.send(:include, I18n::Backend::Fallbacks)
  I18n.load_path, Dir[File.join(settings.root, 'locales', '*.yml')]
  I18n.backend.load_translations
end
```

Now we need to choose the locale that the user wants. There are several solutions (and some can even be mixed together): browser preference, spefic URL, dedicated subdomain, cookies/session management. Only the first three will be shown below:

Browser preference (requires `rack-contrib`) ¶ ↑

```
use Rack::Locale
```

Specific URL ¶ ↑

```
before '/:locale/*' do
  I18n.locale       =       params[:locale]
  request.path_info = '/' + params[:splat ][0]
end
```

Dedicated subdomain ¶ ↑

```
before do
  if (locale = request.host.split('.')[0]) != 'www'
    I18n.locale = locale
  end
end
```

We have all the necessary information to deliver to the user texts/pages in their native language. And for that we will need to select strings and templates according to the desired locale.

Selection of localized strings/objects is easy as it only requires use of standard methods from `I18n`

```
I18n.t(:token)
I18n.l(Time.now)
```

For rendering the templates matching the desired locale, we need to extend the `find_template` method. It needs to select the first template matching the user locale (or at least one acceptable fallback). To help in the selection process templates stored in the `views` directory are suffixed by the name of the locale.

```
helpers do
  def find_template(views, name, engine, &block)
    I18n.fallbacks[I18n.locale].each { |locale|
      super(views, "#{name}.#{locale}", engine, &block) }
    super(views, name, engine, &block)
  end
end
```

Top