

# i18n & l10n: Internationalization and Localization in Rails

Constantin Hofstetter

October 10, 2014

# Definitions

## i18n - Internationalization

Design the application so that it can potentially be adapted to various languages and regions (i18n gem is part of Rails)

## l10n - Localization

The process of adapting internationalized software for a specific region or language by adding locale-specific components and translating text

# Preparations

## Raise an exception on missing translations

```
# config/environments/test.rb
# and
# config/environments/development.rb
Rails.application.configure do |config|
  config.action_view.raise_on_missing_translations = true
end
```

## Instead of I18n.t() use t() shortcut in tests

```
RSpec.configure do |config|
  config.include ActionController::Translation
end
```

See "Foolproof i18n setup on Thoughtbot"

# Best practices: Design

## Make sure things look pretty with a different text length

Other locales will have longer (or shorter) texts: make sure things still look fine.

## Internationalize user interface

Remember that some locales might require text from right to left (e.g. arabic). Also check for user input (e.g. a SlimWiki user from Saudi Arabia).

# Best practices

## Use correct ISO code for locale

Adapt for different regions: e.g. en-US (American English: color)  
vs. en-UK (British English: colour)

See <https://github.com/tigrish/iso> for a comprehensive list

# Pluralization

## Rules

Either add your own `config/plurals.rb` or add gem `'rails-i18n'` to your GEMFILE.

## Localization

```
# de:
#   plurals:
#     elephant:
#       zero: 'Elefant'
#       one: 'Elefant'
#       other: 'Elefanten'
```

```
I18n.locale = :de
I18n.t('plurals.elephant', count: 2) => Elefanten
```

# Interpolation

## Example

```
#  
# en:  
#   hello_user: "Hello %{ name }"  
#  
  
I18n.locale = :en  
I18n.t('hello_user', name: 'Constantin') => "Hello Constantin"
```

# i18n: Front end

## Regular views

Use the `HerbGobbler` gem to add all your plaintext to a yml. You will still need to manually go through all files to add missing strings (e.g. obscure html) and fix interpolations.

## Static javascript

Add the `i18n-js` gem for translation helpers in javascript. Create a custom js file to minimize file size.



# i18n: Database

## ActiveRecord models

Use the Globalize gem to translate ActiveRecord model attributes.

```
# The model
class Post < ActiveRecord::Base
  translates :title, :text
end

# The migration
class AddTranslationsToPosts < ActiveRecord::Migration
  def up
    Post.create_translation_table! :title => :string, :text
      => :text
  end
  def down
    Post.drop_translation_table!
  end
end
```

# Tools

## i18n-tasks

The i18n-tasks gem helps you find and manage missing and unused translations.

```
> i18n-tasks missing
Missing translations (2) | i18n-tasks v0.6.1
+-----+-----+-----+
| Locale | Key          | Details                               |
+-----+-----+-----+
| en     | common.billing_address | app/views/profile/dashboard.html.slim:3 |
| fr     | policy.personal_data   | Never shared, encrypted.               |
+-----+-----+-----+

> i18n-tasks unused
Unused keys (2) | i18n-tasks v0.6.1
+-----+-----+-----+
| Locale | Key          | Value                               |
+-----+-----+-----+
| en     | email.greeting | Hello, {{first_name}}!             |
| fr     | email.greeting | Bonjour {{first_name}}             |
+-----+-----+-----+

> i18n-tasks find '*.billing_address'
1 key matching '*.billing_address' (1 usage) | i18n-tasks v0.6.1
common.billing_address
  app/views/profile/dashboard.html.slim:3 = t 'common.billing_address'

> i18n-tasks translate-missing fr
```

# I10n: Translating the strings

## Use a webservice

Use a webservice such as LocaleApp or WebTranslatelt to get files translated.

## Start early

Start the translation process early - you can work on preparing the frontend and application logic while the files get translated.

# Application Logic

use a before\_action

```
# in ApplicationController
before_action :set_locale

def set_locale
  I18n.locale = case request.host
                 when /\da-z\.-]*\.co\.uk$/
                   "en-UK"
                 else
                   "en-US"
                 end
end
```

## Store preferred locale in User model

If a user chooses a locale, make sure to save it in the database.

# SEO gotchas

## SEO?

Search Engine Optimization: How not to get penalized by Google.

# SEO

## How to determine the locale?

- ▶ by tld e.g. .fr, .de (good!)
- ▶ by path .com/de, .com/fr (ok!)
- ▶ by parameter .com/?ln=de or cookie (bad!)

## What to do with duplicate content?

- ▶ block with robots.txt (not that good)
- ▶ redirect and/or set canonical url (better)

See <https://support.google.com/webmasters/answer/182192>

# SEO

## Sitemap

Use a sitemap to indicate same site is available in different language (resource has to include itself)

```
<url>
  <loc>http://oozou.com</loc>
  <xhtml:link
    rel="alternate"
    hreflang="de"
    href="http://oozou.de"
  />
  <xhtml:link
    rel="alternate"
    hreflang="en"
    href="http://oozou.com"
  />
</url>
```

See <https://support.google.com/webmasters/answer/2620865>

# SEO

## hreflang link tags

Use hreflang for language and regional URLs (resource has to include itself)

```
<head>
<link rel="alternate" href="http://oozou.com" hreflang="en" />
<link rel="alternate" href="http://oozou.th" hreflang="th" />
<link rel="alternate" href="http://oozou.de" hreflang="de" />
<link rel="alternate" href="http://oozou.jp" hreflang="jp" />
</head>
```

See <https://support.google.com/webmasters/answer/189077>



# SEO

## Target countries with different TLDs

Use Google Webmaster tools to target different countries based on your domain.

This only works with gTLDs (generic top-level domain names: e.g. .com, .net) and not with ccTLDs (country-code top-level domain names: e.g. .de, .th).

See <https://support.google.com/webmasters/answer/1347922>

## Analytics

Make sure to track traffic changes in analytics based on countries and locales. Adjust country targeting appropriately.