Crowd-Sourced Translation Engine

Translating your Rails application will never be the same!

http://github.com/berk/tr8n

Presented by Michael Berkovich

TR8N TIMELINE

2007: Geni.com launches

2008: Competing sites launch in local languages around the globe

2009: Geni looks into using I18n and Facebook's solution to translate the site.

Geni uses Google translate to provide minimal solution.

2010: Geni develops tr8n to crowd source translations.

Users translate the site into 10 languages in 4 weeks and international traffic grows to 2/3 of the total traffic.

Yammer uses tr8n Rails Engine Plugin to translate the site into 14 languages in a month.

Geni opens Tr8n Translation Engine to the Rails community in November.

ALTERNATIVE APPROACHES

I18N Framework

- **Pros:** built-in, simple, easy on system resources
- **Cons:** phrase organization, requires translators, no inline translation support, no rules support (context, case)

Facebook Connect

- **Pros:** crowd-sourced, millions of users, supports tokens, rules
- **Cons:** tight Facebook integration, default client-side only (supports server-side integrations, but requires a lot of work)

Google Translate

- **Pros:** easy to integrate, millions of users, uses statistics engine
- **Cons:** client-side only, no rules, translation approximations, many times very sloppy

Web Services (smartling.com, translution.com, and others...)

- **Pros:** server side integration, machine + human translations
- **Cons:** expensive, time consuming, no rules support

CROWD SOURCE BENEFITS

No need to hire professional translators

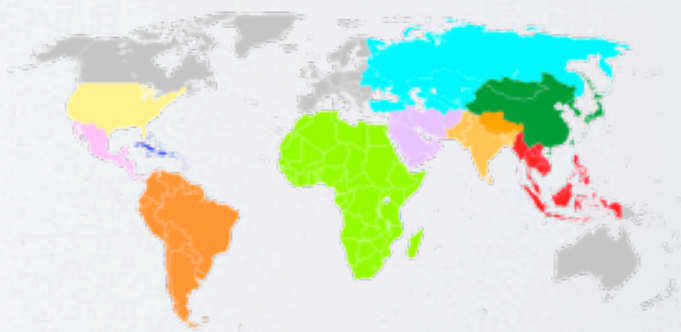
- Your users are free and know your product

Translation is an ongoing process

- As long as you add/change features you will change text
- “Golden Gate Bridge Syndrome”

Minimal cost for adding new languages

- It is easy to enable/add other languages
- Use your users, or Mechanical Turk



And many more...

TR8N FEATURES

- ✔ Support for over 250 languages
- ✔ Crowd-sourced translations
- ✔ No translation yml files!
- ✔ Robust language rules system
- ✔ Intuitive administration tools

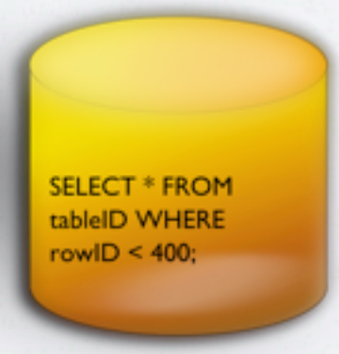
And much more...

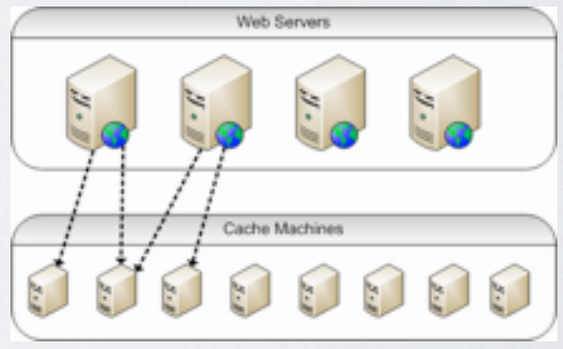
PREREQUISITES

* Rails 2.3.5 to 2.3.10

* MySql, PostgreSQL, SQLite (any database engine will do)

* Memcached (not required, but highly recommended)





Documentation: http://wiki.tr8n.org

INSTALLATION

Third Party Plugins:

```
$ script/plugin install git://github.com/mislav will_paginate.git
$ script/plugin install git://github.com/berk/will_filter.git
$ rake will_filter:sync
$ rake db:migrate
$ cd public; ln -s ../vendor/plugins/will_filter/public/wf wf; cd ..;
```

Tr8n Installation:

```
$ script/plugin install git://github.com/berk/tr8n.git
$ rake tr8n:sync
$ rake db:migrate
$ rake tr8n:init
$ cd public; ln -s ../vendor/plugins/tr8n/public/tr8n tr8n; cd ..;
```

Documentation: http://wiki.tr8n.org

CONFIGURATION

- **Tr8n main configuration files are located here:**
  - config/tr8n/config.yml
  - config/tr8n/site/features.yml
- **Configuration sections:**
  - Features
  - Caching
  - Logger
  - Site Integration
  - Rules Engine
  - API
  - Translators
  - Localization

Documentation: http://wiki.tr8n.org

INTEGRATION

Application Controller and Application Helper

Add the following line to the ApplicationController:

```
include Tr8n::CommonMethods
```

Add the following lines to the ApplicationHelper:

```
include Tr8n::HelperMethods
include Wf::HelperMethods
```

Helper Methods (there are more...)

- **tr8n_scripts_tag**
- **tr8n_language_selector_tag**
- tr8n_translator_login_tag
- **tr8n_language_strip_tag**
- **tr8n_options_for_select**
- tr8n_phrases_link_tag
- tr8n_translator_rank_tag
- tr8n_dir_attribute_tag
- tr8n_splash_screen_tag
- tr8n_language_flag_tag
- tr8n_language_name_tag
- **tr8n_language_table_tag**
- **tr8n_flashes_tag**
- tr8n_client_sdk_scripts_tag

Documentation: http://wiki.tr8n.org

INTERNATIONALIZATION

Tr8n function signature:

```
tr(label, description = "", tokens = {}, options = {})
```

- **label** is a required string
- **description** is an optional, but highly recommended string
- **tokens** is an optional hash of token values - it is required if tokens are used in the label
- **options** is an optional hash of options

Documentation: http://wiki.tr8n.org

FIRST TRANSLATION KEY

```
<%= tr("Hello World") %>
```

or alternatively:

```
<%= "Hello World".translate %>
```

Time for some demo?

Documentation: http://wiki.tr8n.org

ARE THESE THE SAME?

```
<%= tr("Invite", "Link to invite your friends to join the site") %>

<%= tr("Invite", "An invitation you received from your friend") %>
```

- The DESCRIPTION is not mandatory, but it must be used in cases when the label alone is not sufficient enough to determine the meaning of the sentence being translated.

- Tr8n translation engine uses label and description together to create a unique key for each phrase.

- The description serves two purposes:
  - creates a unique key for each label
  - gives a hint to the translators for the context in which the label is used

Documentation: http://wiki.tr8n.org

OTHER TR FLAVORS

```
<%=image_tag(image_url, :alt => trl("Image description")) %>
```

- **trl** - translate label, uses options (:disable_decorations => true)
- **trfn** - translate flash notice message inside a controller
- **trfe** - translate flash error message inside a controller
- **String.translate** or **String.trl** - use anywhere, including models
- **Date.tr** or **Date.trl** - use anywhere, including models
- **Time.tr** or **Time.trl** - use anywhere, including models
- **Array.tro** - translate array elements and join them in a sentence

Documentation: http://wiki.tr8n.org

TR8N TOKENS

There are two main categories of tr8n tokens:

Data Tokens

- Strings surrounded by the curly brackets inside of a label

  {token_ name}

Decoration Tokens

- Strings surrounded by squared brackets inside of a label
- Can contain data tokens inside

  [token_name: value_to_be_decorated]

Documentation: http://wiki.tr8n.org

DATA TOKENS

```
<%= tr("Hello {token}", "Sample", :token => ‘World’) %>

<%= tr("Hello {token}", "Sample", :token => tr(‘World’)) %>
```

You can have nested translations, but be careful no to take partial translations out of the context of the entire sentence.

```
<%= tr("Dear {user}", "Sample", :user => current_user) %>
```

The data tokens are really meant for dynamic data substitution.

Documentation: http://wiki.tr8n.org

DATA TOKENS

```
<%= tr("Dear {user}", "Fragment sample", :user => current_user) %>
```

Token Substitution Options:

- :user => current_user
- :user => [current_user, html_helper(current_user)]
- :user => [current_user, :first_name]
- :user => [current_user, :some_method, "value"]
- :user => [current_user, lambda{|val| html_helper_for(val)}])
- :user => [current_user, lambda{|val, test| html_helper_for(val, test)}], "test"])

Documentation: http://wiki.tr8n.org

DATA TOKEN FLAVORS

There are four flavors of data tokens:

- Simple Data Tokens
- Method Tokens
- Hidden Tokens
- Transform Tokens

Tokens can be extended, if the ones above are not enough... really?

Documentation: http://wiki.tr8n.org

METHOD TOKENS

```
<%= tr("Dear {user.first_name} {user.last_name}", "Fragment sample", :user => current_user) %>
```

- Allows you to call a method on a token itself
- Useful if you have multiple method calls on the same token in one sentence
- Limited to only single form of substitution
- For example, if you want to make only the last_name bold (more on it later)

Documentation: http://wiki.tr8n.org

HIDDEN TOKENS

```
<%= tr("{user} changed {_his_her} name", "Fragment sample", :user => current_user, :_his_her =>
current_user.his_her) %>

<%= tr("you have {count} {_messages}", "Fragment sample", :count => NUM, :_messages =>
"message".pluralize_for(10)) %>
```

- Used primarily for the default language dynamic data substitution that would not make sense in the translated label
- Will not appear as tokens when translator opens the translation dialog
- There is a better approach for the above examples, using the Transform tokens
- There are situations when hidden token can be necessary
- Supports all substitution options

Documentation: http://wiki.tr8n.org

TRANSFORM TOKENS

```
<%= tr("{user} changed {user| his, her} name", "Sample", :user => current_user) %>

<%= tr("You have {count|| message}", "Sample", :count => messages.size) %>

<%= tr("{user| Born on:}", "Sample", :user => current_user) %>

<%= tr("Alex {date| turned, turns, will turn} 25 on {date}", "Sample", :date => some_date) %>

<%= tr("{users|| likes, like} this link", "Sample", :users => [users_list, lambda{|user|
user.first_name}]) %>
```

- Used together with rules defined in the rules engine
- Provide shortcuts for rule based tokens in the site native language
- Token names must be registered as rule enabled tokens (more on it shortly)
- Single pipe means use the label only, double pipe means display the token value followed by the label
- Supports all substitution options

Documentation: http://wiki.tr8n.org

DECORATION TOKENS

```
<%= tr("[link: Click here] to visit our site", "Fragment sample",
    :link => lambda{|text| link_to(text, 'http://www.google.com')}) %>
```

```
<%= tr("[link: Click {url}] to visit our site", "Fragment sample",
       :url => ‘www.google.com’,
       :link => lambda{|text| link_to(text, 'http://www.google.com')}) %>
```

```
<%= tr("[strong: {user}] joined the site", "Fragment sample",
       :user => [current_user, :name] %>
```

Substitution Options:

- :link => lambda{|text| link_to(text, 'http://www.google.com')
- :link => "{$0}"
- :link => ['http://www.google.com']

Documentation: http://wiki.tr8n.org

DEFAULT DECORATIONS

```
<%= tr("[strong: {user}] joined the site", "Fragment sample", :user => [current_user, :name] %>
```

Default Decorations Configuration:

```
config/tr8n/tokens/decorations.yml
```

Default Decorations Examples:

```
strong:     "<strong>{$0}</strong>"
bold:       "<strong>{$0}</strong>"
b:          "<strong>{$0}</strong>"
em:         "<em>{$0}</em>"
italic:     "<i>{$0}</i>"
i:          "<i>{$0}</i>"
link:       "<a href='{$1}' style='{$2}'>{$0}</a>"
func:       "<a href='javascript: {$1}; return false;' style='{$2}'>{$0}</a>"
br:         "<br>{$0}"
```

Documentation: http://wiki.tr8n.org

DEFAULT DATA TOKENS

```
<%= tr("{laquo} previous page", "Fragment sample" %>
```

Default Data Tokens Configuration:

```
config/tr8n/tokens/data.yml
```

Default Data Tokens Examples:

```
ndash:    "&ndash;"   # –
mdash:    "&mdash;"   # —
quot:     "&quot;"    # "
ldquo:    "&ldquo;"   # “
rdquo:    "&rdquo;"   # ”
laquo:    "&laquo;"   # «
raquo:    "&raquo;"   # »
nbsp:     " "    # space
br:       "<br/>"     # line break

etc...
```

Documentation: http://wiki.tr8n.org

TR OPTIONS

```
<%= tr("Добро Пожаловать", "Пример применения", {}, {:locale => ‘ru’, :level => 500}) %>
```

```
<% Tr8n::Config.with_options(:locale => 'ru', :level => 500) do %>
    <%= tr("Добро Пожаловать", "Пример применения") %>
<% end %>
```

- **locale** - specifies the locale of the key
- **level** - specifies minimum translator level who can translate the phrase
- **source** - specifies the location of the phrase, for custom sources
- **skip_decorations** - will not decorate the label even in the translated mode
- **admin** - indicates that the section is part of the admin interface

Documentation: http://wiki.tr8n.org

TR8N RULES ENGINE

```
tr("{actor} gave {target} a present", "Rules sample", :actor => user1, :target => user2)
```

1. The word “**gave**” depends on the gender of the **actor** token = **context rule**
2. The value of the token **target** has to be in a **Dative Form** in many languages = **case rule**

Tr8n supports two types of rules:

- **Language Context Rules** - define context for translations
- **Language Case Rules** - evaluate and modify token values within translations

Documentation: http://wiki.tr8n.org

LANGUAGE CONTEXT RULES

```
tr("You have {count||message}", "Inbox count label", :count => 5)
```

- Transform token is used to make it easy to evaluate the numeric rule for the base language (English in that case)
- In many languages the translated word "messages" depends on the value of the token {count}.
- In languages, like English, there are only two context rules that are necessary to correctly translate the above phrase:
  - **Rule 1**: {count} is 1
  - **Rule 2**: {count} is not 1

Documentation: http://wiki.tr8n.org

LANGUAGE CONTEXT RULES

```
tr("You have {count||message}", "Inbox count label", :count => 5)
```

Translation window presents an option to choose the context rules and provide the translation for each of the rules:

"You have 1 message!"

Context: count is 1

"You have 2 messages!"

Context: count is not 1

Documentation: http://wiki.tr8n.org

LANGUAGE CONTEXT RULES

Tr8n Rules Engine supports the following five default rule types:

- Numeric Rules
- Gender Rules
- List Rules
- Date Rules
- Value Rules

If the above rules are not sufficient, they can be extended in the rules engine

Documentation: http://wiki.tr8n.org

NUMERIC RULES

```
tr("You have {count||message}", "Inbox count label", :count => 5)
```

Numeric Rule Configuration:

```
numeric_rule:
      token_suffixes: [count, num, minutes, hours, days, weeks, months, years]
      object_method:  to_i
```

Numeric Rule Definition:

- Number rule can be simple or complex. You can use "more" and "less" links to use the double conditions for any rule line.
- The first operation options are: "is", "is not", "ends in", "does not end in"
- The linking operator options are: "and", "or"
- The last operation options are: "is", "is not", "ends in", "does not end in"

Documentation: http://wiki.tr8n.org

GENDER RULES

```
tr("Dear {user},", "Reference to a user in a heading of an email", :user => [current_user, :name])

tr("{user} updated {user| his, her} profile.", "Newsfeed story heading", :user => [some_user, :name])

tr("{user| Added On:}", "Label for when the user was added to the site", :user => some_user)
```

Gender Rule Configuration:

```
gender_rule:
    token_suffixes:          [user, profile, actor, target]
    object_method:           gender
    method_values:
      female:                female
      male:                  male
      neutral:               neutral
      unknown:               unknown
```

Gender Rule Definition:

- A rule consists of 2 parts: operator and gender.
- The operator options are: "is", "is not"
- The gender options are: "a male", "a female", "neutral", "unknown"

Documentation: http://wiki.tr8n.org

LIST RULES

```
tr("{user_list|| likes, like} this post.", "Newsfeed story title", :user_list => [[user1], :name])
```

List Rule Configuration:

```
gender_list_rule:
    token_suffixes:          [list]
    object_method:           size
```

List Rule Definition:

- List rule can be simple or complex. You can use "more" and "less" links to use the double conditions for any rule line.
- First operator options: "contains"
- First operator values: "one element", "at least 2 elements"
- Second operator options: "that is", "that is not", "that are", "that are not"
- Second operator values: "male", "female", "unknown", "neutral", "all male", "all female", "of mixed gender"

Documentation: http://wiki.tr8n.org

DATE RULES

```
tr("{user} {birth_date| celebrated, celebrates, will celebrate} {user|his,her} birthday on
{birth_date}!", "Birthday notification", :user => current_user, :birth_date => Date.today)
```

Date Rule Configuration:

```
date_rule:
    token_suffixes:          [date]
    object_method:           to_date
```

Date Rule Definition:

- A rule consists of 2 parts: operator and timeframe.
- The operator options are: "is in the", "is not in the"
- The timeframe options are: "past", "present", "future"

Documentation: http://wiki.tr8n.org

VALUE RULES

```
tr("{actor} is thinking about {target} ", "Fragment sample", :user => current_user)
```

Value Rule Configuration:

```
value_rule:
    token_suffixes:          *
    object_method:           to_s
```

Value Rule Definition:

- A rule consists of 2 parts: operator and value.
- The operator options are: "is", "is not", "starts with", "does not start with", "ends in", "does not end in"
- The value can be anything provided by the language manager

Documentation: http://wiki.tr8n.org

LANGUAGE CASES

```
tr("{actor} gave {target::dat} a present ", "Fragment sample", :user => current_user)
```

Full Date Token Notation:

```
{TOKEN_NAME:TOKEN_DEPENDENCY::LANGUAGE_CASE}
```

```
* A language case is an inflection or use of a noun (or pronoun) to show its
relation to other words in the sentence. In some languages, like Russian, the form
of a token {actor} will changed based on where and how the token is used in a
sentence.

* For instance, in '{target} received a gift from {actor}', {actor} token will be in
the genitive form of the language.

* Language cases can be complex rules with actions that will be executed on the
tokens values if the rule conditions are met.
```

Documentation: http://wiki.tr8n.org

DEMO

RESOURCES

Source: http://github.com/berk/tr8n

Documentation: http://wiki.tr8n.org

Sample Application: http://www.tr8n.org

Questions: michael@geni.com

Want to contribute?

Send me an email and join the project!