

Liquid

moteur de templates pour ruby



Didier Lafforgue

développeur et consultant indépendant

didier@lafforgue.fr



Encore un moteur de templates pour RoR ?



Extrait de Shopify

solutions de boutiques en ligne totalement personnalisables.



Spécificité

- Templates stockés en base en données (ou dans le filesystem).
- Aucun appel système ruby n'est possible.
- Syntaxe à la smarty.
- Utilisable pour tout type de rendu (html, email, ...etc).



Installation

- En gem:

- > sudo gem install liquid

- En plugin (conseillé)

- > cd myapp

- > script/plugin install git://github.com/tobi/liquid.git



Comment ça marche ?



Deux phases

- “Parsing” & compilation:

```
@template = Liquid::Template.parse("Salut {{name}}")
```

- Rendu:

```
@template.render('name' => 'Paul-Emile')
```



Drops

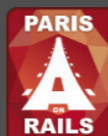
- Syntaxe: `{{ ma_valeur }}`
- Interfaces entre Ruby et Liquid
- Définis dans les modèles eux-mêmes ou bien dans des classes spécifiques.
- Exemple:

Vous avez `{{ cart.size }}` article(s) dans votre panier



Déclarations d'un drop

- ```
class CartDrop < Liquid::Drop
 def initialize(cart)
 @cart = cart
 end
 def size
 @cart.size
 end
end
@template.render('cart' => CartDrop.new(Cart.find_user_id(user)))
```
- ```
class Cart < ActiveRecord::Base
  liquid_methods :size, :total
end
@template.render('cart' => Cart.find_user_id(user))
```
- ```
class Cart < ActiveRecord::Base
 def to_liquid
 { 'size' => size, 'total' => total }
 end
end
@template.render('cart' => Cart.find_user_id(user))
```



# Filtres

- **Syntaxe:**

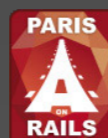
```
Hello {{ 'tobi' | upcase }}
Hello {{ current_user.login | upcase }}
```

- **Utilisation possible de paramètres:**

```
Hello {{ current_user.login | link_to: current_user.url }}
```

- **Chaînage de filtres:**

```
Hello {{ '*tobi*' | textilize | upcase }}
```



# Tags / Blocks

- Syntaxe:  
`{% mon_tag %}`  
ou `{% mon_block %} {% endmon_block %}`
- Conditions (if, unless, case, ...etc), boucles (for), affichage de contenu (include), ...etc.
- Exemple:

```
{% if cart.empty %}
 <p class="empty">empty</p>
{% else %}
 <p>{{ cart.quantity }} item(s)</p>
{% endif %}
```



# Vos propres tags et filtres

si si, c'est possible !!!



# Vos propres filtres

- Avec un module:

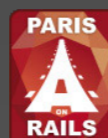
```
module LiquidStuff::MyFilters
 def mon_textilize(input)
 RedCloth.new(input).to_html
 end
end
```

- Enregistrement nécessaire:

```
Liquid::Template.register_filter(LiquidStuff::MyFilters)
```

- Utilisation:

```
Hello {{ '*bla bla*' | mon_textilize }}
```



# Vos propres tags

- Héritage des classes Liquid::Block ou Liquid::Tag

```
class RoundedCorners < Liquid::Block
 def initialize(tag_name, markup, tokens)
 ...
 end
 def render(context)
 ...
 end
end
Liquid::Template.register_tag('rounded_corners', RoundedCorners)
```

- Utilisation:

```
{% rounded_corners 'main' %}
<h1>My Shop</h1>
<p>bla bla bla</p>
{% endrounded_corners %}
```



# Ce que j'ai oublié de dire

- Le contexte (registers, ...etc)
- Les erreurs (utilisation de render!)
- Les urls (comment les générer ?)



# Best practices Liquid

exemple d'un (petit) CMS



# Démo & Sources

- Pour la démo:
  - front: <http://liquidcms.herokuapp.com/>
  - back: <http://liquidcms.herokuapp.com/admin/>
- Les sources:
  - <http://liquidcms.herokuapp.com/liquidcms.zip>



# Avant de commencer

- Table liquid\_resources: *name, layout\_id, path, text, type*.
- 3 sous-modèles: layout, page et snippet.
- champ *text* contient le template liquid.
- Table assets: *slug, ...etc*.



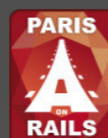
# Rendu d'une page

- En appliquant les exemples donnés par les tutoriaux:

```
def show
 @page = Page.find_by_path(params[:url])
 template = Liquid::Template.parse(@page.text)
 content = template.render({ 'page' => @page }, :registers =>
{ 'controller' => self })

 template = Liquid::Template.parse(@page.layout.text)
 html = template.render({ 'content_for_layout' => content, 'page' =>
@page }, :registers => { 'controller' => self })

 response.headers["Content-Type"] = 'text/html; charset=utf-8'
 render :text => html
end
```



Nous pouvons faire  
mieux



# Simplement

```
def show
 render :page => params[:url]
end
```



# Et plus encore !

```
GET /products/1
def show
 @product = Product.find(params[:id])
 render :page => '/product'
end
```

```
<h1>{{ product.title }}</h1>
<p>Price: {{ product.price | number_to_currency }}</p>
<p>Quantity: {{ product.quantity }}</p>
```



# Et les layouts ?



# Un exemple

```
<html>
 <head>
 <title>My great website</title>
 </head>
 <body>
 <h1>{% assets.logo | image_tag %} Company name</h1>
 <div id="content">
 {{ content_for_layout }}
 </div>
 {% include 'footer' %}
 </body>
</html>
```



# Layout avancé

- Page avec Liquid et layout avec ERB, c'est possible !

```
GET /tests/hello_world
def hello_world
 @title = 'Hello world'
 render :page => '/hello_world', :layout => 'simple'
end
```

- L'instruction *yield* dans le layout insérera le contenu de la page.



# Les snippets

- Exemple du footer dans le layout
- Utilisation du tag `{% include %}`

```
{% include 'menu' %}
```

- Templates Liquid en base de données

```
Liquid::Template.file_system = Liquid::DbFileSystem.new
```



# Snippets avec ERB ?

pourquoi faire ?



# Un exemple

```
<html>
<head>
 <title>Bla bla</title>
</head>
<body>
 <%= render :snippet => 'ads' %>
 <div id="header">
 <h1>My ecommerce website</h1>
 </div>
 <div id="content">
 <%= yield %>
 </div>
</body>
</html>
```



# Images (1/2)

- Modèle Asset

```
class Asset < ActiveRecord::Base
 ...
 def to_liquid
 Liquid::AssetDrop.new(self)
 end
end
```

- Drop Asset

```
class AssetDrop < BaseDrop
 ...
 def to_s
 self.path
 end
end
```



# Images (2/2)

- Drop “librairie” d’assets

```
class AssetsDrop < Drop
 def before_method(meth)
 Asset.find_by_slug(meth.to_s)
 end
end
```

- Utilisation `{{ assets.<asset_slug> }}`

```
<div id="banner">{{ assets.banner | image_tag }}</div>
```



# La lenteur de Liquid



# Faux problème

Solution: Rails cache



# Les ressources sur Internet



- **Le site officiel:**  
<http://www.liquidmarkup.org/>
- **Le wiki:**  
<http://github.com/tobi/liquid/wikis>
- **Le groupe de discussions:**  
<http://groups.google.com/group/liquid-templates>
- **Le forum de Shopify:**  
<http://forums.shopify.com/categories/2>



Merci de votre  
attention



# Des questions ?

trop timide ? [didier@lafforgue.fr](mailto:didier@lafforgue.fr)

