ERB Hacks

seki@ruby-lang.org

ERBすごいぞ！という自慢をするので、わたしの承認欲求を満たして欲しい

ERB Hacks

seki@ruby-lang.org

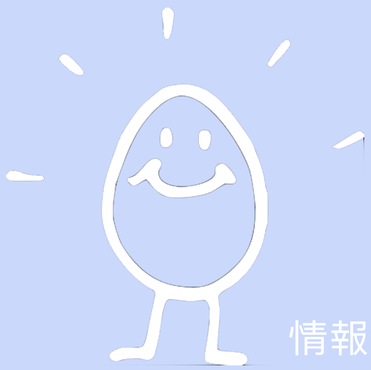
○●● わたしについて

- dRuby/Rinda/ERBを書きました
- ふだんはtoRubyというコミュニティにいます
- ポケモンカードWCS2010栃木県代表です

島根大の講義、楽しかった！

toRuby

- 「とちぎRubyの勉強会」は来月200回
- 池澤さんの「Rubyの家庭教師をお願いしたいです」のメールが発端
- 池澤さんにお仕事を依頼すると、もれなくtoRubyの技術支援がついてくる！お得！



情報デザイン・ロゴ

おそらく、矢板市で一番Herokuを使ってると思う


ポケモンカードゲーム
トレーナーズウェブサイト
「古代」「未来」の
ポケモン、襲来！
10月27日(金)発売！
https://www.pokemon-card.com/ex/sv4/

今日の話

- 古代編
  - ERB予言とその答え合わせの話

- 未来編
  - ブロックを使ったhelperが実装できそうな話

ERBすごいぞ！という自慢をするので、わたしの承認欲求を満たして欲しい

○●● 古代編

- ERB予言とその答え合わせ

1999年頃のお話です

ERB前史（1999年）

- HTMLのエンティティ表記を置換するライブラリを書いてたら、eRubyというものを作っていることを教えてもらった

```
<H2>検索結果</H2>
[&KEYS;]
<UL>
&PAGES;
</UL>
</DIV>
<ADDRESS><A href="mailto:&EMAIL;">&AUTHOR;</A></ADDRESS>
</BODY>
</HTML>
EOF
    doc
  end

  needle = {
    'HIDDEN' => '',
    'KEYS' => 'hello ruby',
    'PAGES' => '<LI>empty',
    'AUTHOR' => 'Masatoshi SEKI',
    'EMAIL' => 'm_seki@mva•biglobe.ne.jp'
  }
```

エンティティ表記ならWebオーサリングツールの邪魔しないだろう、と想像した（使ったことないけど）

ruby-dev:5286 (1999-02-19) Re: htmlelem.rb (Re: HTML generator)

eRubyの仕様書

アルバイトの前田さん!!

ePerl

printと <%= .. %>が同じ!?

いま開発中なんだな～

```
From: Shugo Maeda <shugo@netlab•co.jp>
To: ruby-dev@netlab•co.jp
Subject: [ruby-dev:5376] Re: eRuby (Re: htmlelem.rb)
Date: Sat, 20 Feb 1999 03:53:35 +0900   [thread overview]
Message-ID: <199902191853.DAA21807@po.aianet.ne.jp> (raw)
In-Reply-To: Wakou Aoyama's message of "Fri, 19 Feb 1999 21:27:34 +0900"

前田です。

Wakou Aoyama <wakou@fsinet•or.jp> writes:

> ePerl 情報。
>
> という程のものではありませんが、ePerl はあまり一般的ではないと思われる
> ので、どんなものか少し紹介。(私もあまり知らないのですが)

実はeRubyの仕様書(？)がすでに手元にあったりします。

1. <% print "Hello World!" %>
2. <%= "Hello World!" >

のいずれも出力はHello World!

3. <%# ... %>

コメント

# これは実装するのは僕ではないです。(たぶん)

--
前田 修吾
```

ePerlの影響を強くうけているみたいだぞ

○●● 作ってみたい

- 制御構造を文書に埋め込めるのいいなー
- 標準出力っていうのはCGIに特化しすぎでは...

CGIのおさらい

- CGIでレスポンスは$stdoutへの印字
- このためCGIの後半はHTMLを印字するputsが並ぶ
- eRubyのような仕様を思いつくのは自然かも

古のMacはまた違ったインターフェイスだったかも（AppleScriptで書けたと思う）

退屈なputs

```
puts %Q{<!DOCTYPE html>}
puts %Q{<html lang="ja">}
puts %Q{<head>}
puts %Q{</head>}
puts %Q{<body>}
puts %Q{<ul>}
ENV.each do |row|
  puts %Q!<li>#{row[0]}</li>!
end
puts %Q{</ul>}
puts %Q{</body>}
puts %Q{</html>}
```


古来のCGIはだいたいこんな感じ

here document使うとマシ

```
puts %Q{<!DOCTYPE html>}
puts %Q{<html lang="ja">}
puts %Q{<head>}
puts %Q{</head>}
puts %Q{<body>}
puts %Q{<ul>}
ENV.each do |row|
  puts %Q!<li>#{row[0]}</li>!
end
puts %Q{</ul>}
puts %Q{</body>}
puts %Q{</html>}
```

```
puts <<EOS
<!DOCTYPE html>
<html lang="ja">
<head>
</head>
<body>
<ul>
EOS
ENV.each do |row|
  puts %Q!<li>#{row[0]}</li>!
end
puts <<EOS
</ul>
</body>
</html>
EOS
```

putsの回数は減らせた

putsからeRubyへ

```
puts <<EOS
<!DOCTYPE html>
<html lang="ja">
<head>
</head>
<body>
<ul>
EOS
ENV.each do |row|
  puts %Q!<li>#{row[0]}</li>!
end
puts <<EOS
</ul>
</body>
</html>
EOS
```

```
<!DOCTYPE html>
<html lang="ja">
<head>
</head>
<body>
<ul>
<% ENV.each do |row| %>
  <li><%= row[0] %></li>
<% end %>
</ul>
</body>
</html>
```


Stringの式展開では制御構造が書きにくい（書ける？）のだがeRubyならかんたん

この仕様はよくない気がする

- CGIの最終段の処理に特化し過ぎてる
- このスタイルは近いうちに変わる予感がする
  - ページの生成はこんなに単純ではなくなるぞ
  - CGIのモデルはサーバーぽいものへ変わるよ、きっと

古いUNIX, X11などでの開発経験からそう感じた、のかな...

単純じゃなくなるぞ予言

- ページの一部をメソッドで生成したくなると思う

```
<h2>report hoge</h2>
<ul>
<% result.each do |row| %>
  <li><%= format_date(row.date) %></li>
<% end %>
</ul>
```

```
def format_date(date)
  date.strftime("%Y-%m-%d")
end
```


Helper?

単純じゃなくなるぞ予言

- テンプレートを入れ子にしたくなる
  - 共通のヘッダ、ナビケーション、フッタとコンテンツ

```
<%= header %>
<%= navi %>
<%= content %>
<%= footer %>
```

CGIのモデルは変わる 予言

- CGIのプロセスのモデルは変わりそう
- 前処理のコストやリクエスト間の情報のやりとりなどから、なんらかの長生きなプロセスが必要になる
- 並行処理で$stdoutは問題になると思う

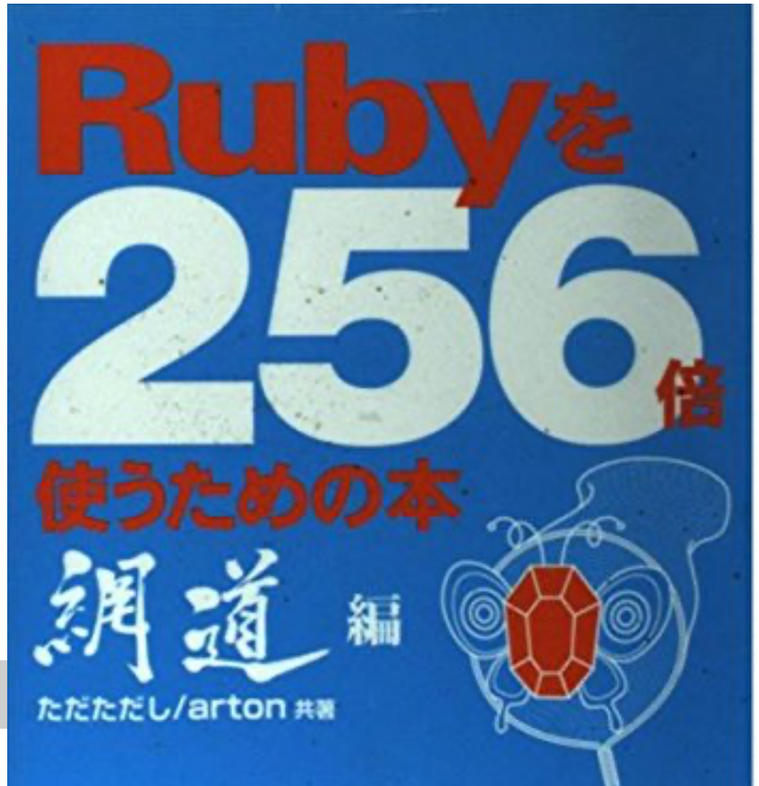
ERbLight

- eRubyのように動くERbと、予言に合わせた仕様のERbLightを用意した
  - 印字ではなく文字列を組み立てる
  - eRuby表記でRubyのアプリを書く(のを支援する)
    - 変換器として働き、任意のbindingでのevalや、メソッド化を可能に

議論して揉めるべきだったがシャイなのでモノで表明した

Rubyを256倍使うための本

- この仕様の可能性を誰か気がつくといいなあ
  - 気づかれた！→網道編



たださんとartonさんがeRuby/dRubyを絶賛するすごい本

ERbLightからERBへ

- ERBに改名されRubyの標準添付ライブラリになった

○●● 四半世紀が過ぎた

- 予言、だいぶいい線いったのでは！
- ERBが影響したライブラリが存在するのがうれしい
  - 楽しいので自分用のライブラリは自分で書いてる

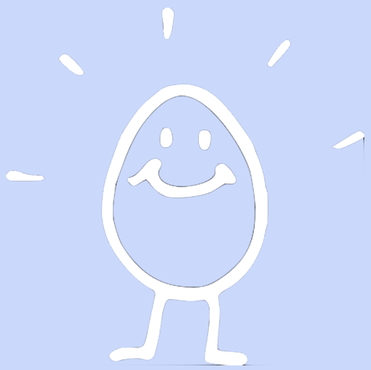
Merb・Railsは関係ある、と思っていいよね

未来編

- Railsのブロックを使ったhelperが実装できそうな話
  - 10年前にあきらめたブロック問題

もとのネタ

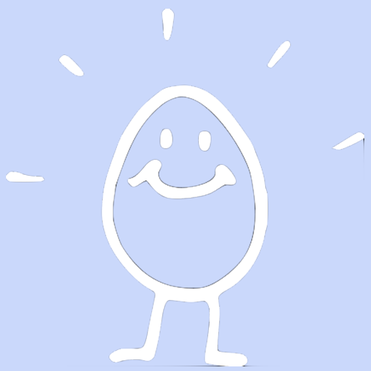
- 池澤プロダクツで多用されているテクニック
  - ERB#def_methodによるメソッド化
  - テンプレート内でのProcの利用



↑toRubyの技術支援がついてるのでお得な池澤さんの「情報デザイン」のロゴ

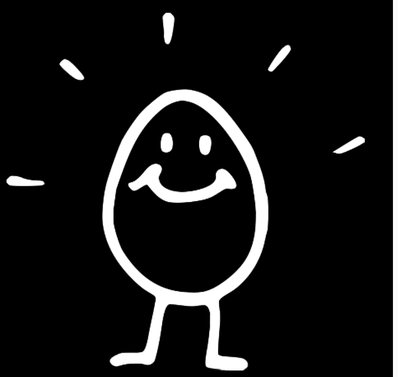
Procの利用

- ユースケース：似たような設問が繰り返し出てくるアンケートのWebページを作る
  - 質問ごとに大きな表組みの回答欄がある
  - 要素のidやformの名前はユニークにしたい
  - できればひとつのテンプレートの中で書きたい



↑矢板市北部で一番Herokuを使ってる「情報デザイン」のロゴ

実際のアンケートのようす



20アンケート　ヘルプ　施設の

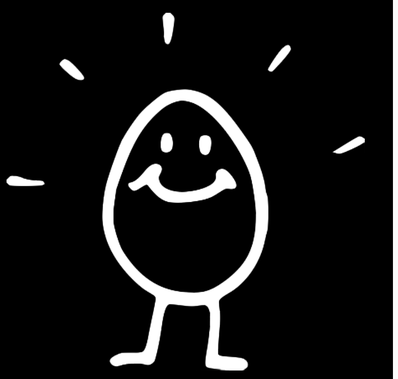
I.一般項目

- 貴施設に関する設問 未回答
- 作業療法士の労働環境に関する設問 未回答
- 作業療法部門の管理・運営・リスク管理等に関する設問 未回答
- 地方自治体等が運営する会議・事業への参画状況に関する設問 未回答

II.作業療法士の臨床活動に関する設問

- 臨床活動の領域に関する設問 完了
- 医療保険（共通） 未回答
- 医療保険（身体障害） 回答中
- 医療保険（精神障害） 回答中
- 医療保険（発達障害） 未回答
- 介護保険 回答不要
- 障害福祉 回答不要
- 教育関連（特別支援学校など） 回答不要
- 職業関連 回答不要

質問ごとに表組の回答欄がある...と思うでしょ



臨床活動の領域に関する設問 完了

医療保険（共通） 未回答

医療保険（身体障害） 回答中

医療保険における作業療法「身体障害領域」に関する設問

※該当する選択肢がない場合は、「該当なし」またはそれに相当する選択肢を選んでください。
※人数などで該当がない項目にはゼロ(0)を入力してください。

2021年10月7日または指定日の臨床活動についてお答えください

（10月7日でない場合、10月8日 ）

問.50

10月7日または指定日に身体障害領域の作業療法に従事した**作業療法士の人数**をお答えください。 3 人

問.51

10月7日または指定日に身体障害領域の作業療法を実施した**対象者の人数**をお答えください。 人 人

問.52

10月7日または指定日に身体障害領域の作業療法を実施した**対象者の人数を下記の年齢区分ごと**にお答えください。

15歳以下	人	人
16歳～64歳	人	人
65歳～74歳	人	人
75歳以上	人	人

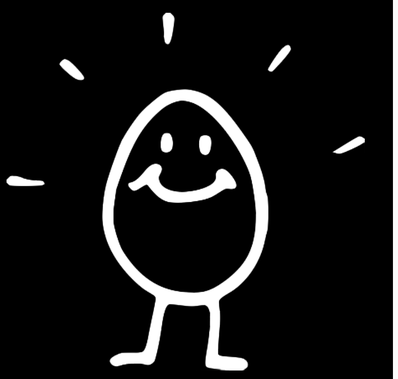
実は各セルごとに詳細の表がある

	外来	編集		外来	編集
	訪問	編集		訪問	編集
生活期	入院	編集	終末期	入院	編集
	外来	編集		外来	編集
	訪問	編集		訪問	編集

問.60

10月7日または指定日に実施した**実施した作業療法の種目**を、**最多10項目（順不同）**まで《資料４》より選んでチェックしてください。
※《資料４》の表示／非表示は「編集」ボタンのクリックで

急性期	入院	編集	回復期	入院	編集
	外来	編集		外来	編集
	訪問	編集		訪問	編集
生活期	入院	編集	終末期	入院	編集
	外来	編集		外来	編集
	訪問	編集		訪問	編集



問.61

上記種目から、リハビリテーションチームの中で作業療法士が**特に実施すべき作業療法種目**は何であると考えますか。**5項目（順不同）**を《資料４》より選んでチェックしてください。
※《資料４》の表示／非表示は「編集」ボタンのクリックで

急性期	入院	編集	回復期	入院	編集

Procの利用

- Procで処理の一部を使いまわせる

```
<% t = Proc.new do |arg| %>
<h3><%= arg %>回答欄</h3>
<table id="table-<%= arg %>">
  ...
</table>
<% end %>

<h3>質問1</h3>
最初の質問文だよ
<% t.call("q1") %>

<h3>質問2</h3>
二つ目の質問だよ
<% t.call("q2") %>
```

ここを再利用したい

ここで呼び出す

```
<h3>質問1</h3>
最初の質問文だよ

<h3>q1回答欄</h3>
<table id="table-q1">
  ...
</table>


<h3>質問2</h3>
二つ目の質問だよ

<h3>q2回答欄</h3>
<table id="table-q2">
  ...
</table>
```

Procの利用

- Procで処理の一部を使いまわせる

```
<% t = Proc.new do |arg| %>
<h3><%= arg %>回答欄</h3>
<table id="table-<%= arg %>">
  ...
</table>
<% end %>

<h3>質問1</h3>
最初の質問文だよ
<% t.call("q1") %>

<h3>質問2</h3>
二つ目の質問だよ
<% t.call("q2") %>
```

HTML片を組み立てる処理
そのものがProcになる

パラメータ渡せる

Procの利用

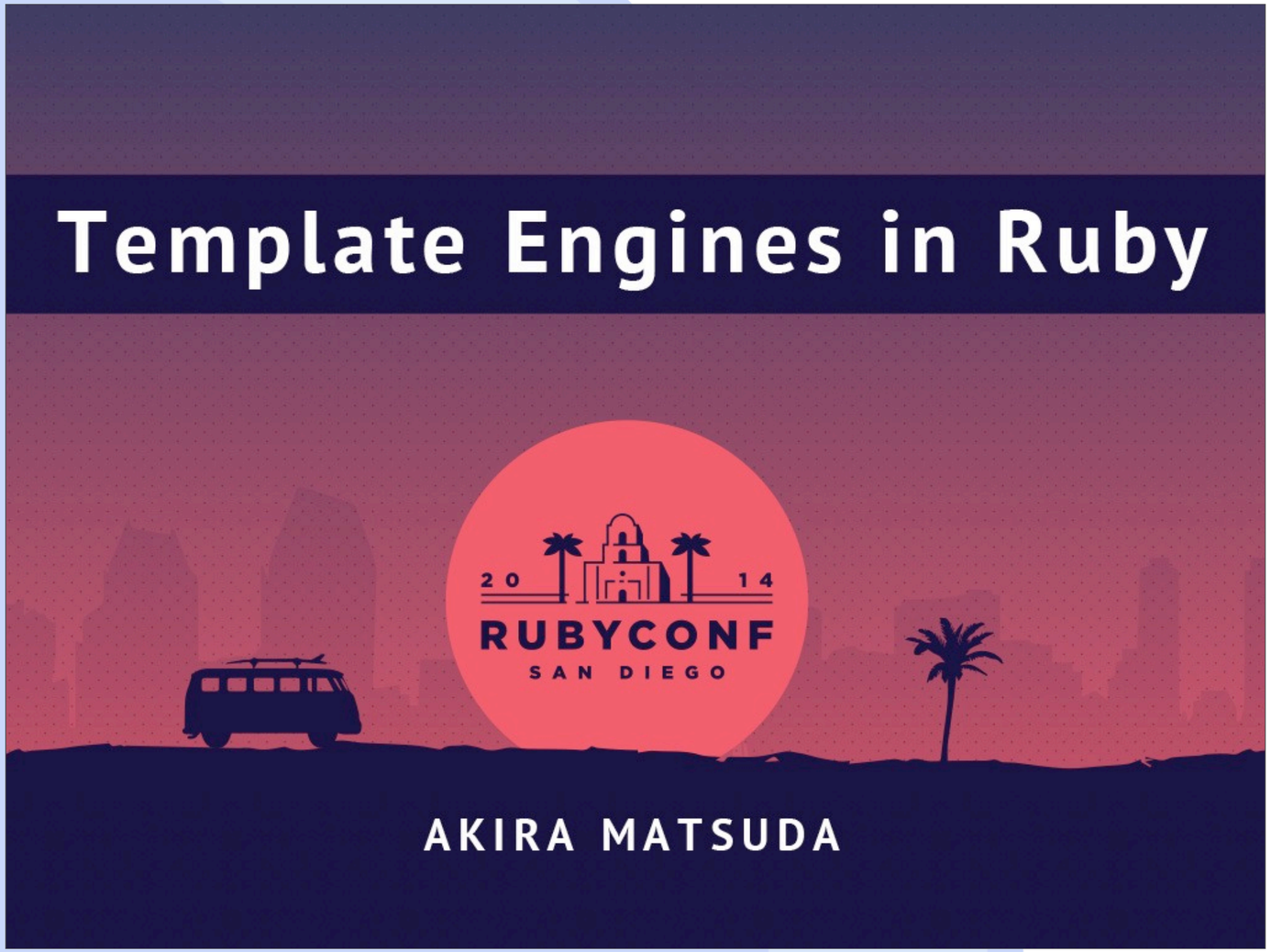
- Procで処理の一部を使いまわせる

```
<% t = Proc.new do |arg| %>
<h3><%= arg %>回答欄</h3>
<table id="table-<%= arg %>">
  ...
</table>
<% end %>

<h3>質問1</h3>
最初の質問文だよ
<% t.call("q1") %>

<% t.call("q2") %>
```

<%= ... %>ではないので注意

あれ？なんかこれ見たことある...

松田さんの2014年のトーク



Ruby Confね

10年ほど前に相談された

- RailsのテンプレートエンジンをERBに戻せる？
  - 現在はErubisなのかな
- 技術的な問題はなさそうだったので試しました

できませんでした

But gave up.

- Because he didn't like this part in ActionView + Erubis
- BLOCK_EXPR = /\s+(do|\{)(\s*\|[^|]*\|)?\s*\Z/
- This code scans the template with the Regexp and detects the Ruby block, but this kind of code could be imperfect
- So this is not acceptable as an ERB spec, said Seki-san.

Railsに正規表現でブロックの開始を検知して処理を切り替えるコードがあった

これが解けなかった

That is why this syntax is not acceptable in ERB

```
<%= form_for @article do |f| %>

  ...

<% end %>
```

Railsに正規表現でブロックの開始を検知して処理を切り替えるコードがあった

<%= ... %>とブロック

- ブロック付きメソッドを <%= ... %>に置くのが難しい

```
<%= form_with do %>
  ...
<% end %>
```

<%= ... %>とブロック

- こうなっちゃう

```
<%= form_with do %>
  ...
<% end %>
```

```
#coding:UTF-8
_erbout = +''; _erbout.<<(( form_with do ).to_s); _erbout.<< "\n  ...
\n".freeze

;  end ; _erbout.<< "\n".freeze
; _erbout
```

SyntaxError

あれ？なんかできそうかも💡

- RWCの応募したあとにProcの追試しながら考えてたら解けそうな気がしてきた

https://gist.github.com/seki/610a42932a85209aaa33547ae983bbdf

なおしかた

- ((...).to_s)の括弧がじゃま

```
<%= form_with do %>
  ...
<% end %>
```

```
#coding:UTF-8
_erbout = +''; _erbout.<<(( form_with do ).to_s); _erbout.<< "\n  ...
\n".freeze

;  end ; _erbout.<< "\n".freeze
; _erbout
```

((...).to_s) をなくせばよいかも？

→ 結果バッファクラスに移動させる

なおしかた

- Rubyへの変換方法はカスタマイズできる

```
<%= form_with do %>
```

一時変数の名前と初期化

式の連結
((...).to_s)をなくせばOK

リテラルの連結

```
#coding:UTF-8
_erbout = +''; _erbout.<<(( form_with do ).to_s); _erbout.<< "\n  ...
\n".freeze

;  end ; _erbout.<< "\n".freeze
; _erbout
```

結果の取得

Rubyむずかしい

- そんなに単純じゃなかった

```
_erbout << h("str")                                  # OK

_erbout << h "str"                                   # SyntaxError
```

<<と括弧なしのメソッド呼び出し

```
_erbout.concat h "str"                               # OK but ....
_erbout.concat form_with "arg" do |arg|              # NG
  ...                                                #
end                                                  # concatにブロックが渡る
```

<<でなくふつうのメソッド呼び出しにしてもこのブロックはconcatのブロック引数になる

```
_erbout = h "str"
_erbout = form_with "arg" do |arg|
  ...
end
```

代入なら大丈夫なのに...

代入しながらメソッド呼べばよい?

- += でうまくいった

```
_erbout += h "str"                                  # OK
_erbout += form_with "arg" do |arg|                 # OK
  ...
end
```

+=すばらしい

ERBOut

- 結果バッファクラスの追加
- いろんなトリックがある
  - +で自分自身を返して+=をだます
  - captureのためのしかけ

```
class ERBOut
  Buffer = String # SafeBuffer if rails

  def initialize(s='')
    @str = Buffer.new(s)
  end

  def to_s
    @str
  end

  def <<(other)
    @str << other
  end

  def +(other)
    @str << other.to_s
    self
  end

  def capture(*arg, &block)
    save = @str
    @str = Buffer.new
    yield(*arg)
    return @str
  ensure
    @str = save
  end
end
```

式の連結処理の一部をここへ移動

自分自身を返す

ERBOut

- 結果バッファクラスの追加
- いろんなトリックがある
  - +で自分自身を返して+=をだます
  - captureのためのしかけ

```
class ERBOut
  Buffer = String # SafeBuffer if rails

  def initialize(s='')
    @str = Buffer.new(s)
  end

  def to_s
    @str
  end

  def <<(other)
    @str << other
  end

  def +(other)
    @str << other.to_s
    self
  end

  def capture(*arg) block)
    save = @str
    @str = Buffer.new
    yield(*arg)
    return @str
  ensure
    @str = save
  end
end
```

もとのバッファを退避
一時バッファに差し替え

一時バッファをreturn
もとのバッファを復元

なんかできちゃった

- 中間結果のためのERBOutクラスも使ったらできた

```
<%= form_with do %>
  .
<% e
```

一時変数の名前と初期化

ここはブロックの内側だよ！

```
#coding:UTF-8
_erbout = ERB::ERBOut.new; _erbout +=   form_with do  ; _erbout <<
"\n  ...\n".freeze

;  end ; _erbout <<  "\n".freeze
; _erbout.to_s
```

結果の取得

式の連結
+= がミソ

https://gist.github.com/seki/610a42932a85209aaa33547ae983bbdf

captureの実装

- ブロックのローカル変数をさわってcaptureのしかけを呼び出す

```
def capture(*arg, &block)
  block.binding.local_variable_get(:_erbout).capture(*arg, &block)
end
```

このブロックのbindingの _erbout

```
<% it = capture do %>
  ...
<% end %>
```

今後

- Rails環境でも動くことを確認できるといいなあ
  - めんどうなのでだれかやってくれるといいなあ

（うごいてもうごかなくても沖縄で報告したい）

今日の話

- 古代編
  - ERB予言とその答え合わせの話

- 未来編
  - ブロックを使ったhelperが実装できそうな話