用Erlang快速开发Web快速开发框架

TL;WL

```
git clone
git://github.com/bhuztez/
razor
```

不撸兔子……，他的梗又好玩又多，只是没开发，什么……失败人士……

——古伦木与欧巴

失败人士	99%	开发效率
成功人士	1%	运行效率

Web Framework Benchmarks

Framework ? Disable all

actframework	activeweb	aiohttp	akka-http	aleph	API Star	api_hour	ASP.NET	asp.net-mvc
aspnetcore	beego	beyondj	blaze	bottle	cakephp	chicagoboss	clancatsframewo	codeigniter
Collie	colossus	compojure	cpoll-cppsp	curacao	cutelyst	cygnite	dancer	django
dropwizard	EntityFrameworl	evhttp-sharp	express	falcon	falcore	Fat Free	ffead-cpp	finagle
finatra	finch	fintrospect	flask	fuel	gemini	gin	go	Goji
grails	grape	grizzly	hapi	Hexagon	hhvm	Hot	http4s	hyper
iron	jawn	Jersey	jester	jooby	JSONNet	kami	kelp	kemal
Klein	koa	kohana	ktor	lapis	laravel	Limonade	lithium	Lumen
luminus	mojolicious	nancy	netty	ngx_mruby	nickel	ninja	None	Octopus
officefloor	openresty	padrino	pedestal	phalcon	Phoenix	phpixie	phreeze	plack
play1	play2	pyramid	rack	rack-sequel	rails	rapidoid	rapidoid-http-fa	redstone
rest-express	Revenj	RevenjJVM	ringo	roda-sequel	rouille	s-server	scruffy	Servant
servicestack	silex	silicon	sinatra	sinatra-sequel	slim	snap	spark	Spock
spray	spring	start	stream	symfony2	tapestry	tokio-minihttp	treefrog	TurboGears
ULib	urweb	Vapor	vertx-web	vibe.d	vibed	web-simple	web2py	webgo
Webware	weppy	wheezy.web	wicket	wt	yaf	yesod	yii2	Zend
ZendFramework:								

GeniusVczh
@geniusvczh

Following

Shopping is hard, let's reinvent the wheel!

Retweets **1.0K**
Likes **65K**

5:14 PM - 01 Jul 2010

21 1.0K 65K

Why Erlang?

- maximum Erlang DM throughput is approximately double the C++ DM throughput
- pure Erlang implementation is approximately three times faster
- Erlang DM uses 150% more memory
- without human intervention the Erlang DM recovers after load drops
- pure Erlang DM is 1/7th of the size

—— High-level Distribution for the Rapid Production of Robust Telecoms Software: Comparing C++ and Erlang

开发效率来源

- small language
  - pattern matching
  - 函数
  - 消息
- 热更新
- parse_transform

pattern matching (1)

```
1 = 1
a = a
a = 'a'
{1,2} = {1,2}
[1,2] = [1|[2]]
[1,2] = [1,2|[]]
```

匹配

```
1 = 2
a = aa
a = 1
{1,2} = {1,3}
{1,1} = {1}
[1,2] = [1|[]]
```

不匹配

pattern matching (2)

```
1> 1 = X.
* 1: variable 'X' is unbound
2> X = 1.
1
3> X = 2.
** exception error: no match of
right hand side value 2
```

相当于 1 = 2

pattern matching (3)

```
{X,X} = {1,1}
[1|_] = [1,2,3]

#{a:=1}=#{a=>1}

#{a:=1}
   = #{a=>1,b=>2}
```

```
{X,X} = {1,2}
[1,2|_] = [1]

#{a:=1}=#{a=>2}

#{a:=1,b:=2}
   = #{a=>1}
```

函数 (1)

```
-module(hello).
-export([test/0]).
hello(world) -> ok;
hello(_) -> error.
test() ->
    ok = hello(world),
    error = hello(0),
    ok.
```

函数 (2)

```
1> c(hello).
{ok,hello}
2> hello:test().
ok
3>
```

消息

```
1> Self = self().
<0.33.0>
2> spawn(fun() -> Self ! hello end).
<0.36.0>
3> receive Msg -> Msg end.
hello
4>
```

C-c C-k

```
send_event(Conn) ->
    receive
    after 2000 ->
      send_chunk(Conn,
    <<"data: stay tuned.\r\n\r\n">>)
    end,
    ?MODULE:send_event(Conn).
```

parse_transform (1)

```
-module(example).

-compile({parse_transform,
          razor_id_trans}).
```

parse_transform (2)

```
-module(razor_id_trans).
-export([parse_transform/2]).

parse_transform(Forms, _Options) ->
    Forms.
```

parse_transform (3)

```
-module(razor_id_trans).
-export([parse_transform/2]).

{attribute,1,module,razor_id_trans}
{attribute,2,export,
 [{parse_transform,2}]}
```

parse_transform (4)

```
parse_transform(Forms, _Options) ->
    Forms.
{function,4,parse_transform,2,
 [{clause,4,
   [{var,4,'Forms'},
    {var,4,'_Options'}],
   [],
   [{var,5,'Forms'}]}]}
```

parse_transform (5)

```
{error,{Line,Module,Descriptor}}
{warning,{Line,Module,Descriptor}}

Module:format_error(Descriptor)
```

razor

- Demo Driven Development
  - razor_url_dispatch
    - razor_peg
  - razor_db
- 尽可能少的依赖
  - epgsql
- 编译期检查，及早发现错误

razor_url_dispatch

- 编译时生成(tagged) DFA代码
- 展开UTF8编码以及URL编码
- 检查同一URL会不会被两条规则匹配
- 展示Erlang的优势
  - attribute
  - parse_transform

razor_url_example (1)

```
-pattern({integer,"[0-9]+",
{erlang,binary_to_integer},
{erlang,integer_to_binary}}).
-dispatch({root,"posts/",
           {dispatch, post}}).
-dispatch({post,"",
           {endpoint,index}}).
-dispatch({post,"{id:integer}",
           {endpoint, post}}).
```

razor_url_example (2)

```
{index,#{},<<"a">>} =
    url_dispatch(<<"/posts/?a">>),
{post,#{id:=10},<<>>} =
    url_dispatch(<<"/posts/10">>),
{post,#{id:=10},<<>>} =
    url_dispatch(<<"/posts/1%30">>),
error =
    url_dispatch(<<"/posts/1%3">>),
```

razor_url_example (3)

```
[<<"/">>,<<"posts/">>]
    = url_reverse(index,
                  #{}),

[<<"/">>,<<"posts/">>,<<"10">>]
    = url_reverse(post,
                  #{id => 10}),
```

razor_url_example (4)

```
url_dispatch(Bin) ->
  case url_dfa(Bin) of
    error -> error;
    {18, #{}, Rest} -> {index, #{}, Rest};
    {23, #{1 := T1, 2 := T2}, Rest} ->
      {post,
        #{id =>
            erlang:binary_to_integer(
              razor_url:decode(
                binary:part(Bin,T1,T2-T1)))},
       Rest}
  end.
```

razor_url_example (5)

```
url_reverse(post, #{id := Vid}) ->
  [<<"/">>, <<"posts/">>,
   razor_url:encode(
     erlang:integer_to_binary(Vid))
  ];
url_reverse(index, #{}) ->
  [<<"/">>, <<"posts/">>].
```

razor_url_example (6)

```
-dispatch({post,"0",
           {endpoint,index}}).
-dispatch({post,"{id:integer}",
           {endpoint, post}}).

rule conflict
```

razor_peg

- 自带的leex/yecc不支持unicode
- 类似PEG，一匹配就忽略后面的
- 只改变语义，不改变语法
- 展示Erlang的优势
  - pattern matching

razor_peg_example (1)

```
test() ->
    {ok, 1, ""} = int("1"),
    {ok, 12, ""} = int("12"),
    {ok, 12, "a"} = int("12a"),
    error = int("a"),
    error = int(""),
    ok.
```

razor_peg_example (2)

```
int(S) -> int(0,S).
-rule(int/2).
int(Acc, S) ->
    {ok, N, S1} = digit(S),
    int(Acc*10+N, S1);
int(Acc, S) ->
    {ok, N, S1} = digit(S),
    {ok, Acc*10+N, S1}.
```

razor_peg_example (3)

```
-rule(digit/1).
digit([H|T])
  when H >= $0, H =< $9 ->
    {ok, H - $0, T}.
```

razor_peg_example (4)

```
int(Acc,S)->{ok,N,S1}=digit(S),
              int(Acc*10+N, S1)
case {V1, V2} of
  {V0Acc, V0S} ->
    case digit(V0S) of
      {ok, V0N, V0S1} ->
        int(V0Acc * 10 + V0N, V0S1);
      _ -> error end end
```

razor_db

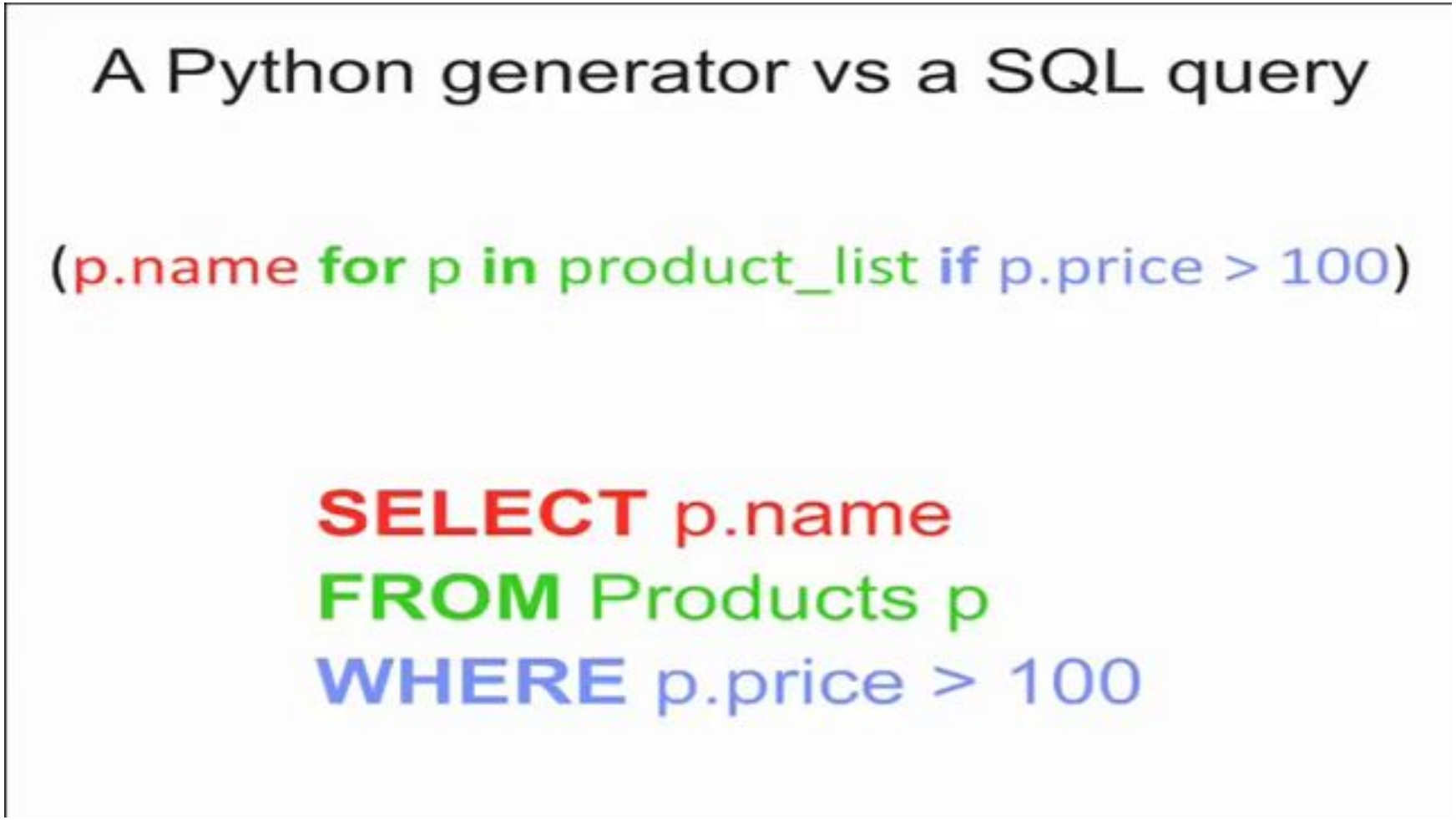
- 将List Comprehension转换成SQL
  - 支持GROUP BY和Aggregation
  - 支持子查询
  - 支持(Recursive) Common Table Expression
- 编译时警告没有用到的参数和输出
- 展示Erlang的优势
  - pattern matching
  - erl_syntax_lib:annotate_bindings/2

How Pony ORM translates Python generators to SQL queries

Python generator to SQL translation

1. Decompile bytecode and restore AST
2. Translate AST to ‘abstract SQL’
3. Translate ‘abstract SQL’ to a specific SQL dialect

How Pony ORM translates Python generators to SQL queries



razor_db_examples (1)

```
razor_db:select(DB,
   [ Name
     || #{name := Name, price := Price}
           <- from(product_list),
        Price > 100 ])
[ Name || {Name} <-
            razor_db:query(DB,
"SELECT T1.name FROM product_list AS T1 "
"WHERE (T1.price >100)",
                           [])]
```

razor_db_examples (2)

```
[ {Name, Count}
  || #{id := ID, name := Name} <- from(items),
group_by(Name), Count<-[count(ID)],
Count == X])
[ {Name, Count}
  || {Count, Name} <- razor_db:query(DB,
"SELECT count(T1.id), T1.name FROM "
"items AS T1 GROUP BY T1.name "
"HAVING (count(T1.id) = $1)",
                                   [X])]
```

razor_db_examples (3)

```
razor_db:select(
  DB, [ A || #{id := A} <- from(i1),
exists(
select([#{}||#{id:=B}<-from(i2),
                A == B]))])
[ A || {A} <- razor_db:query(DB,
"SELECT T1.id FROM i1 AS T1 WHERE "
"(EXISTS((SELECT 1 FROM i2 AS T2 "
"WHERE (T1.id = T2.id))))",
                        [])]
```

razor_db_examples (4)

```
razor_db:select(DB,
  #{i => [[#{id => ID}
          || #{id := ID}
             <- from(items)]]},
  [ ID || #{id := ID} <- from(i) ])
[ ID || {ID} <- razor_db:query(DB,
"WITH RECURSIVE i(id) AS ((SELECT "
"T1.id AS id FROM items AS T1))"
" SELECT T2.id FROM i AS T2",
                          [])];
```

razor_db_examples (5)

```
razor_db:select(
  DB,
  [ ID
    || #{id := ID, name := Name}
          <- from(item) ])

Warning: variable 'Name' is unused.
```

razor_db_examples (6)

```
razor_db:select(DB,
[ ID
  || #{id := ID} <- from(i1),
     exists(
       select([#{}||#{id := U}
                    <- from(i2)]))]
)
Warning: variable 'U' is unused.
```

razor_db_examples (7)

```
razor_db:select(
  DB,
  [ ID
    || #{id := ID} <- from(items),
       _ <- [param(1)]]);

Warning: param is unused.
```

razor_db_examples (8)

```
razor_db:select(
  DB, [ #{post_id => PID, content => Content,
          vote_id => VID, is_upvote => IsUpvote}
        || #{id := PID, content := Content} <- from(post),
           #{id := VID, is_upvote := IsUpvote}
             <- join(vote,
                     c(post_id) == PID,
                     c(user_id) == UID)
      ]).
```

razor_db_examples (9)

```
[ #{post_id => PID, content => Content,
    vote_id => VID, is_upvote => IsUpvote}
  || {Content, IsUpvote, PID, VID}
        <- razor_db:query(DB,
"SELECT T1.content, T2.is_upvote, T1.id, T2.id FROM post AS T1 "
"LEFT OUTER JOIN vote AS T2 "
"ON (T2.post_id = T1.id) AND "
"(T2.user_id = $1)",
                          [UID])]
```

razor_api_example

```
PUT /1 "Hello, world!"
OK
PUT /2 "Hello, world!"
OK
GET /
1
2
GET /1
Hello, world!
```

razor_api_example (1)

```
-module(razor_api_example).
-compile({parse_transform,
          razor_url_dispatch}).
-compile({parse_transform,
          razor_db}).
-export([start/0, url_dispatch/1,
url_reverse/2, handle_request/3,
middlewares/0, not_found/0]).
```

razor_api_example (2)

```
-pattern({integer,"[0-9]+",
  {erlang, binary_to_integer},
  {erlang, integer_to_binary}}).
-dispatch({root, "",
           {endpoint, index}}).
-dispatch({root, "{id:integer}",
           {endpoint, item}}).
```

razor_api_example (3)

```
start() -> start(8000).
start(Port) ->
  razor_http_server:start(
    Port, [], ?MODULE, []).
connect_db() ->
  epgsql:connect(
    "127.0.0.1", "razor", "",
    [{database, "razor"}]).
```

razor_api_example (4)

```
middlewares() -> [].
not_found() ->
  response(404, <<"Not Found">>).
response(Code, Body) ->
  { http_response, Code,
    <<"Content-Type: text/plain; "
      "charset=utf-8\r\n">>,
    Body}.
```

razor_api_example (5)

```
handle_request('GET',index,_) ->
  with_db(fun(DB) ->
    Index = razor_db:select(DB,
[ io_lib:format("~w~n", [ID])
  || #{id := ID} <- from(razor_item)
]),
    response(200, Index)
  end);
```

razor_api_example (6)

```
Index =
  [ io_lib:format("~w~n", [ID])
    || {ID}
         <- razor_db:query(DB,
"SELECT T1.id "
"FROM razor_item AS T1",
   [])]
```

razor_api_example (7)

```
handle_request('GET',item,#{id:=X})->
  with_db(fun(DB) ->
      case razor_db:select(DB,
        [Data||#{id:=ID,data:=Data}
                  <- from(razor_item),
                ID == X]) of
          [Data]->response(200,Data);
          [] -> not_found()
    end
  end);
```

razor_api_example (8)

```
with_db(Fun) ->
  {ok, DB} = connect_db(),
  try
    {ok, _, _}
        = epgsql:squery(DB,"BEGIN"),
    Fun(DB)
  after
      epgsql:close(DB)
  end.
```

就是这样