Ruby off Rails

by Stoyan Zhekov
Kyoto, 17-May-2008

自己紹介

- 名前:　ストヤン　ジェコフ
- ブルガリア人
- ３人男の子のパパ

内容

- Coupling and Cohesion
- ウェブアプリに必要なこと
- ORM - Sequel (簡単な見解)
- Ruby ウェブエボリューション
- Enter Rack - The Middle Man
- Thin, Ramaze, Tamanegi
- まとめ, その他, 質問

Coupling and Cohesion (1)

- Cohesion - 凝集度 (ぎょうしゅうど)

  http://en.wikipedia.org/wiki/Cohesion_(computer_science)
- Coupling - 結合度 (けつごうど)

  http://en.wikipedia.org/wiki/Coupling_(computer_science)

Coupling and Cohesion (2)

正しいウェブアプリの組み立て方

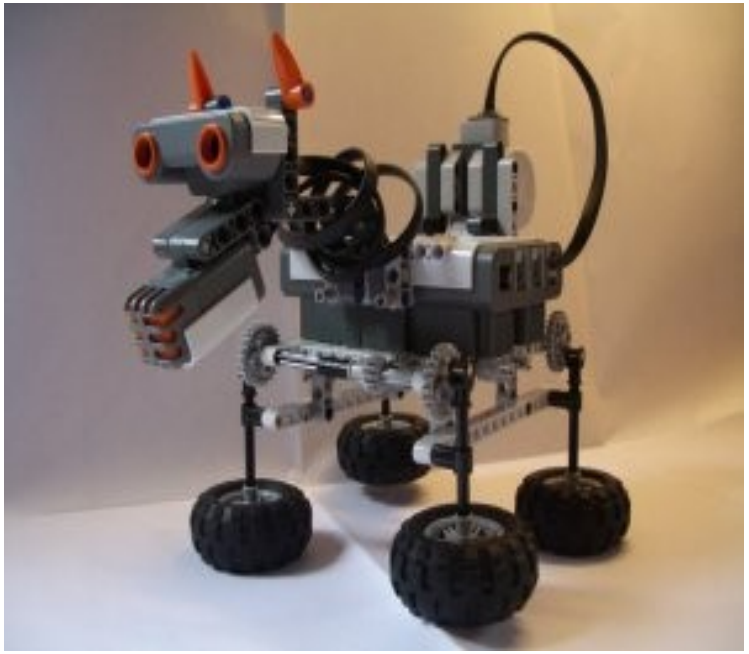
- **High cohesion**
  - 少ない動作でたかい機能性
- **Low (loosely) coupling**
  - 簡単な入れ替え

Coupling and Cohesion (3)

Google vs Amazon

Coupling and Cohesion (5)

vs



Railsについて

Ruby on Rails

- 全部セットでよかった。しかし、今は？
- Convention over configuration – いいのか?
- DRY – 可能?
  - Generators
  - Plugins
  - Libraries

Railsなしの人生

できる?

ウェブアプリに必要なこと

- データベースの使い方: ORM
- Request手順: CGI env, アップロード
- ルーティング: URLとクラスの組み合わせ
- Rendering (views): rendering libraries
- Sessions: persist objects or data

Database ORM

- Active Record ?= ActiveRecord
- ActiveRecord – たくさんの問題点
  - http://datamapper.org/why.html
- Data Mapper – 早いけどまだ未完
  - まだコンポジットキーなし
  - 自分のDBドライバーが必要
- Sequel – よし。気に入ってる。

ORM – Sequel (なぜ?)

- Pure Ruby
- Thread safe
- Connection pooling
- DB queries構築のためのDSL
- ObjectとDBレコードの組み合わせの軽いORM
- Transactions with rollback

ORM – Sequel (Core)

- データベースのつなげ方

```
require 'sequel'
DB = Sequel.open 'sqlite:///blog.db'


DB = Sequel.open 'postgres://cico:12345@localhost:5432/mydb'


DB = Sequel.open("postgres://postgres:postgres@localhost/my_db",
  :max_connections => 10, :logger => Logger.new('log/db.log'))
```

ORM – Sequel (Core, 2)

```
DB << "CREATE TABLE users (name VARCHAR(255) NOT NULL)"

DB.fetch("SELECT name FROM users") do |row|
  p r[:name]
end


dataset = DB[:managers].where(:salary => 50..100).order(:name, :department)


paginated = dataset.paginate(1, 10) # first page, 10 rows per page
paginated.page_count #=> number of pages in dataset
paginated.current_page #=> 1
```

ORM – Sequel (Model)

```
class Post < Sequel::Model(:my_posts)
  set_primary_key [:category, :title]

  belongs_to :author
  has_many :comments
  has_and_belongs_to_many :tags

  after_create do
    set(:created_at => Time.now)
  end

end
```

ORM – Sequel (Model, 2)

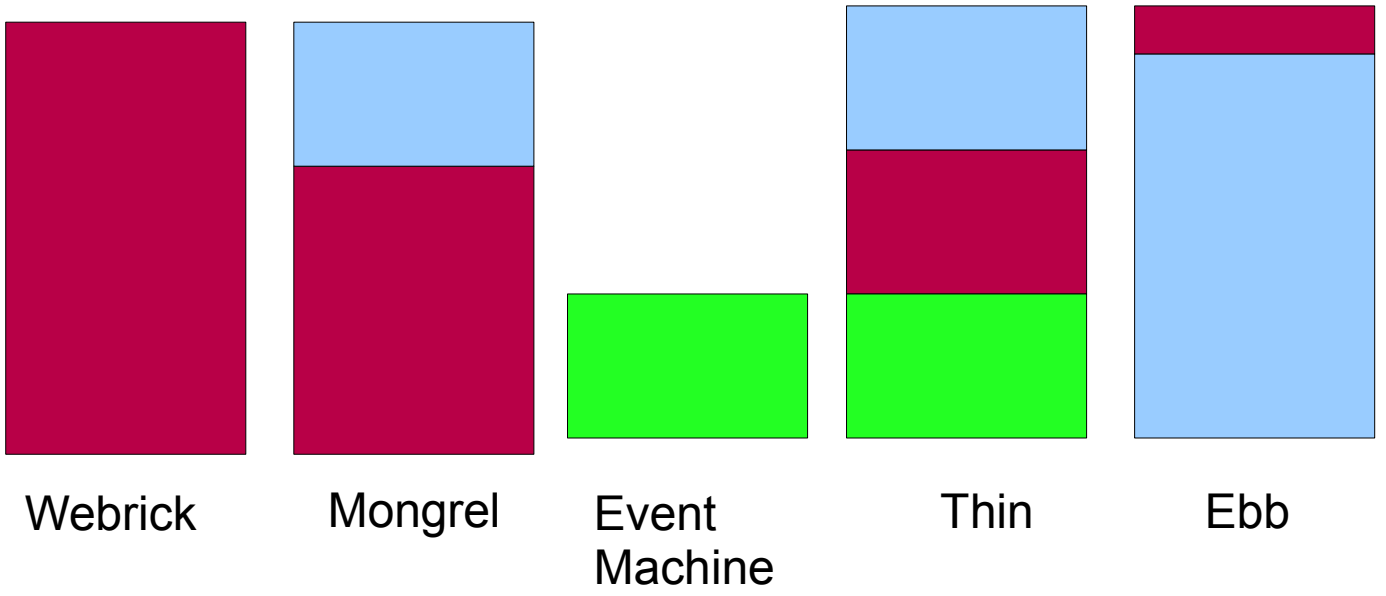
```
class Person < Sequel::Model
    has_many :posts, :eager=>[:tags]

    set_schema do
        primary_key :id
        text :name
        text :email
        foreign_key :team_id, :table => :teams
    end

end
```

Evolution

Ruby Web Evolution



Thin ウェブサーバー

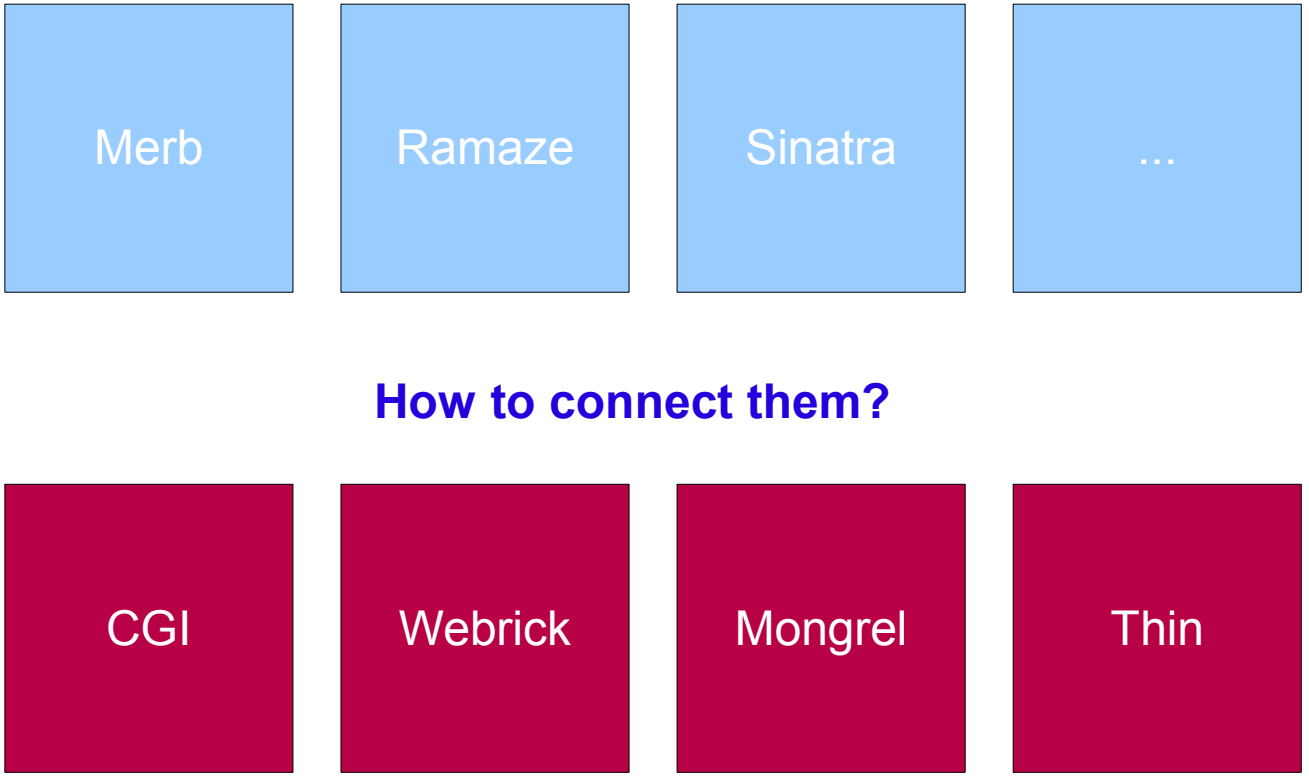
- 早い – Ragel and Event Machine
- クラスタリング
- Unix sockets - nginx
- Rack-based – Rails, Ramaze etc.
- Rackup ファイルサポート (後で)

Thin + Nginx

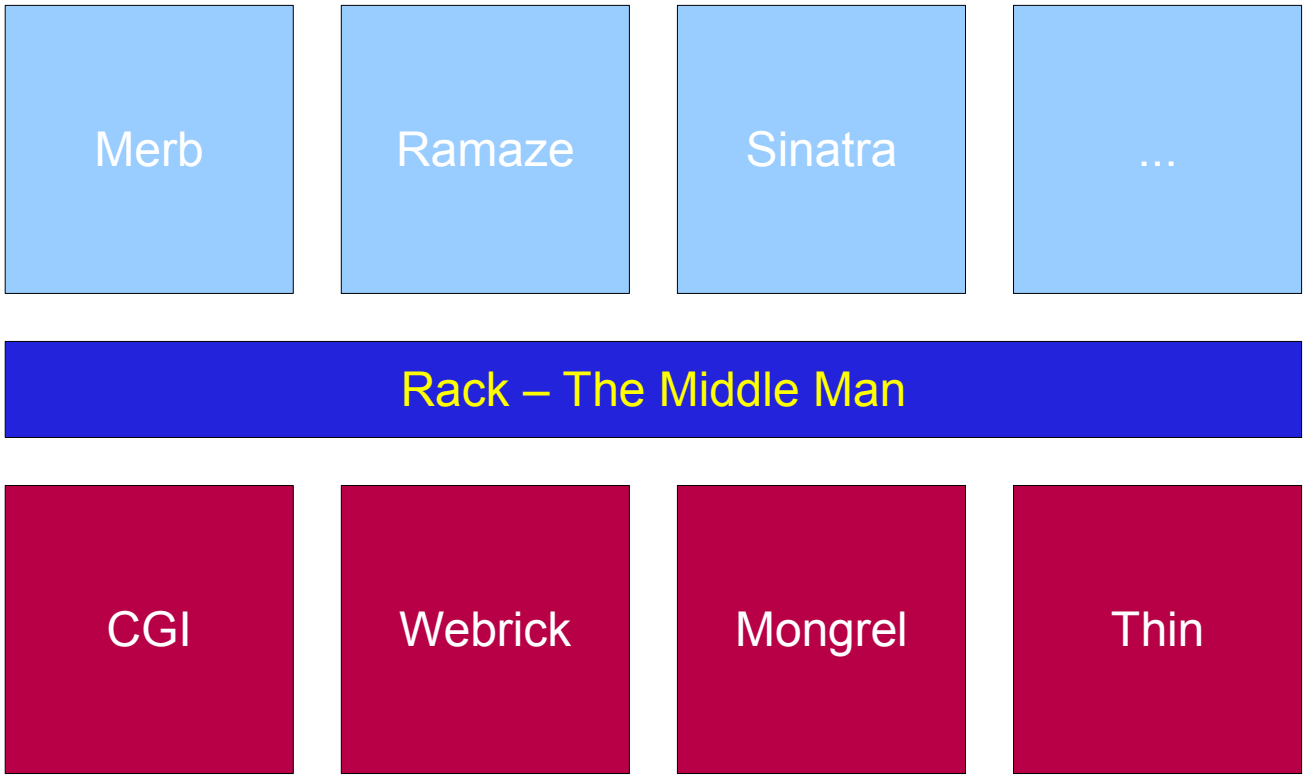
```
thin start --servers 3 --socket /tmp/thin
thin start --servers 3 --port 3000

#  nginx.conf
upstream  backend {
   fair;
   server   unix:/tmp/thin.0.sock;
   server   unix:/tmp/thin.1.sock;
   server   unix:/tmp/thin.2.sock;
}
```

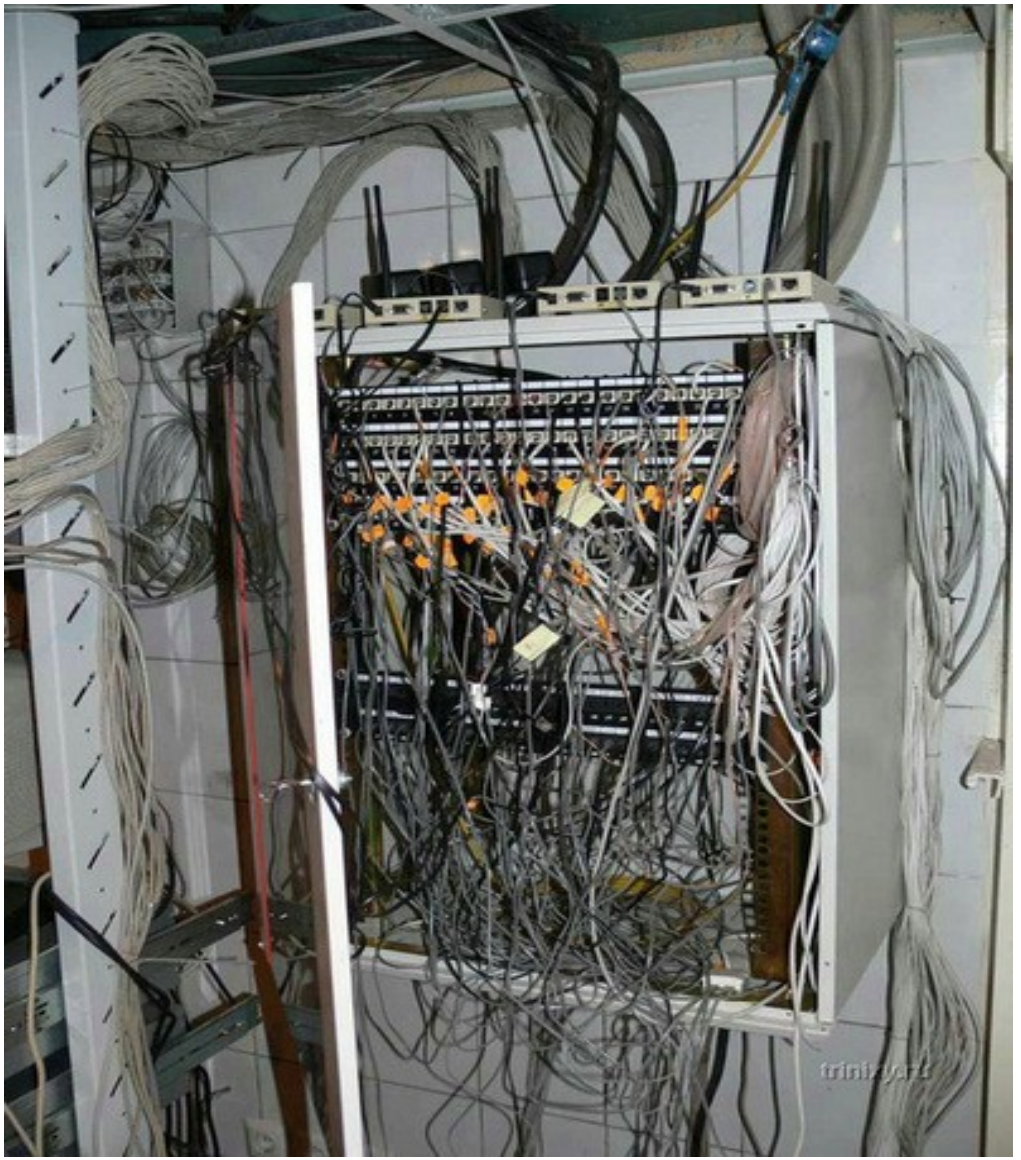
問題点


Merb
Ramaze
Sinatra
...
How to connect them?
CGI
Webrick
Mongrel
Thin

ソリューション - Rack



Rackって何?



Rackって何なの?

http://rack.rubyforge.org/

By Christian Neukirchen

Rackって何？

- HTTPっぽいもののRuby APIのための簡単な仕様(とその実装)
  - Request -> Response
- callメソッドを持ち、envを受け取り以下を返すようなオブジェクト:
  - a status, i.e. *200*
  - the headers, i.e. *{‘Content-Type’ => ‘text/html’}*
  - an object that responds to **each**: *‘some string’*

これがRackだ!

```
class RackApp
    def call(env)
        [ 200, {"Content-Type" => "text/plain"}, "Hi!"]
    end
end

app = RackApp.new
```

Free Hugs

http://www.freehugscampaign.org/

Hugs ウェブアプリ (1)

p 'Hug!'

ruby hugs.rb

```
%w(rubygems rack).each { |dep| require dep }

class HugsApp
    def call(env)
        [ 200, {"Content-Type" => "text/plain"}, "Hug!"]
    end
end

Rack::Handler::Mongrel.run(HugsApp.new, :Port => 3000)
```

Rack::Builder

```
require 'hugs'
app = Rack::Builder.new {
  use Rack::Static, :urls => ["/css", "/images"], :root => "public"
  use Rack::CommonLogger
  use Rack::ShowExceptions

  map "/" do
    run lambda { [200, {"Content-Type" => "text/plain"}, ["Hi!"]] }
  end

  map "/hug" do
    run HugsApp.new
  end
}

Rack::Handler::Thin.run app, :Port => 3000
```

Rack Building Blocks

- request = Rack::Request.new(env)
  - request.get?
- response = Rack::Response.new('Hi!')
  - response.set_cookie('sess-id', 'abcde')
- Rack::URLMap
- Rack::Session
- Rack::Auth::Basic
- Rack::File

Rack Middleware

自分のuse Rack::「...」ブロックを作れる?

```
class RackMiddlewareExample
  def initialize app
    @app = app
  end

  def call env
    @app.call(env)
  end
end
```

ウェブアプリに必要なこと

- データベースの使い方(ORM): Sequel
- Request手順: Rack::Request(env)
- ルーティング: Rack 'map' and 'run'
- Rendering/views: Tenjin, Amrita2
- Sessions: Rack::Session

Does it scale?

rackup hugs.ru

```
require 'hugs'
app = Rack::Builder.new {
use Rack::Static, :urls => ["/css", "/images"], :root => "public"

map "/" do
  run lambda { [200, {"Content-Type" => "text/plain"}, ["Hi!"]] }
end

map "/hug" do
  run HugsApp.new
end
}
Rack::Handler::Thin.run app, :Port => 3000
```

Hugs サービス

- rackup -s webrick -p 3000 hugs.ru
- **thin start --servers 3 -p 3000 -R hugs.ru**
- SCGI
- FastCGI
- Litespeed
- ....

Rack-based フレームワークス

- Invisible – thin-based フレームワーク in 35 LOC
  - http://github.com/macournoyer/invisible/
- Coset – REST on Rack
- Halcyon – JSON Server Framework
- Merb
- Sinatra
- Ramaze
- Rails !?

RailsにもRackが必要

- RailsはRackをすでに持っている、しかし:
  - raw req -> rack env -> Rack::Request ->CGIWrapper -> CgiRequest
- Ezra Zygmuntowicz (merb)'s repo on github
  - http://github.com/ezmobius/rails
  - raw req -> rack env -> ActionController::RackRequest
  - *./script/rackup -s thin -c 5 -e production*

Ramaze

- 何？: モジュラーウェブアプリ フレームワーク
- どこで？: http://ramaze.net/
- どうやって？: gem install ramaze
- git clone http://github.com/manveru/ramaze.git
- 誰が？: Michael Fellinger and Co.

なぜ Ramaze?

- モジュラー (low [loosely] coupling)
- Rack-based
- 簡単な使い方
- たくさんの例 (~25) – facebook, rapaste
- フリースタイルな開発
- フレンドリーなコミュ - #ramaze
- “All bugs are fixed within 48 hours of reporting.”

Deployment オプションズ

- CGI
- FastCGI
- LiteSpeed
- Mongrel
- SCGI
- Webrick
- Ebb
- Evented Mongrel
- Swiftiplied Mongrel
- Thin
- Rack-basedなら使える

Templating engines

- Amrita2
- Builder
- Erubis
- Ezamar
- Haml
- Liquid
- Markaby
- RedCloth
- Remarkably
- Sass
- Tagz
- Tenjin
- XSLT

Ezamar

```
<div class="userlist">
  <?r @users.each do |user| ?>
    #{user.name}
  <?r end ?>
</div>
```

本当に簡単?

```
%w(rubygems ramaze).each {|dep| require dep}

class MainController < Ramaze::Controller
  def index; "Hi" end
end

class HugsController < Ramaze::Controller
  map '/hug'
  def index; "Hug!" end
end

Ramaze.start :adapter => :thin, :port => 3000
```

Ramaze使い方

```
class SmileyController < Ramaze::Controller
    map '/smile'

    helper :smiley

    def index
        smiley(':)')
    end
end
```

Ramaze 使い方 (2)

```
module Ramaze
    module Helper
        module SmileyHelper
            FACES = {
              ':)' => '/images/smile.png',
              ';)' => '/images/twink.png'}
            REGEXP = Regexp.union(*FACES.keys)

            def smiley(string)
                string.gsub(REGEXP) { FACES[$1] }
            end
        end
    end
end
```

他のStack

- ORM – Sequel
- ウェブサーバー – Thin + NginX
- Rack Middleware
- Ramaze フレームワーク
- Templates – HAML, Tenjin, Amrita2

まとめ

- Railsを使うなとは言いません
- オープンマインドで
- SequelはORMに適している
- Rackを使え! – DIY framework, DIY server
- Ramazeはかっこよくて、使いやすい!