Mechanize

shameless content grabbing

Content grabbing

Content grabbing

Profit

Content grabbing

Profit

Morale

Content grabbing

Profit

Morale

This is not my damn business.

Deprecations

0.9.x	1.0
WWW::Mechanize	Mechanize

Examples

```ruby
@browser = Mechanize.new

page = @browser.get('http://mega.genn.org')

page.search('h2 a.entry-title').each do |a|
  post = Post.new

  post.title = a.text
  post.orig_url = a[:href]

  post.save
end
```

```ruby
@browser = Mechanize.new

page = @browser.get('http://mega.genn.org')

page.search('h2 a.entry-title').each do |a|
  post = Post.new

  post.title = a.text
  post.orig_url = a[:href]

  post_page = @browser.click(a)
  post.content = post_page.at('.content .entry-content').to_s

  post.save
end
```

Pulling all together

In the beginning...

```
def initialize
  @browser = Mechanize.new
end
```

Dealing with <meta>s...

```
def initialize
  @browser = Mechanize.new {|a| a.follow_meta_refresh = true }
end
```

Conserving memory...

```
def initialize
  @browser = Mechanize.new {|a| a.follow_meta_refresh = true }
  @browser.max_history = 2
end
```

Simulating user agent...

```
def initialize
  @browser = Mechanize.new {|a| a.follow_meta_refresh = true }
  @browser.max_history = 2
  @browser.user_agent_alias = 'Linux Mozilla'
end
```

Using tor anonymizer...

```
def initialize
  @browser = Mechanize.new {|a| a.follow_meta_refresh = true }
  @browser.max_history = 2
  @browser.user_agent_alias = 'Linux Mozilla'
  @browser.set_proxy '127.0.0.1', '8118'
end
```

Setting cookies...

```
def initialize
  @browser = Mechanize.new {|a| a.follow_meta_refresh = true }
  @browser.max_history = 2
  @browser.user_agent_alias = 'Linux Mozilla'
  @browser.set_proxy '127.0.0.1', '8118'
  set_some_cookies
end

def set_some_cookies
  cookie = Mechanize::Cookie.new 'is_bot', "advanced one"
  cookie.path = '/'
  cookie.domain = 'mega.genn.org'

  uri = URI::HTTP.build( :host => 'mega.genn.org', :path => '/')
  @browser.cookie_jar.add uri, cookie
end
```

What to do with JS?

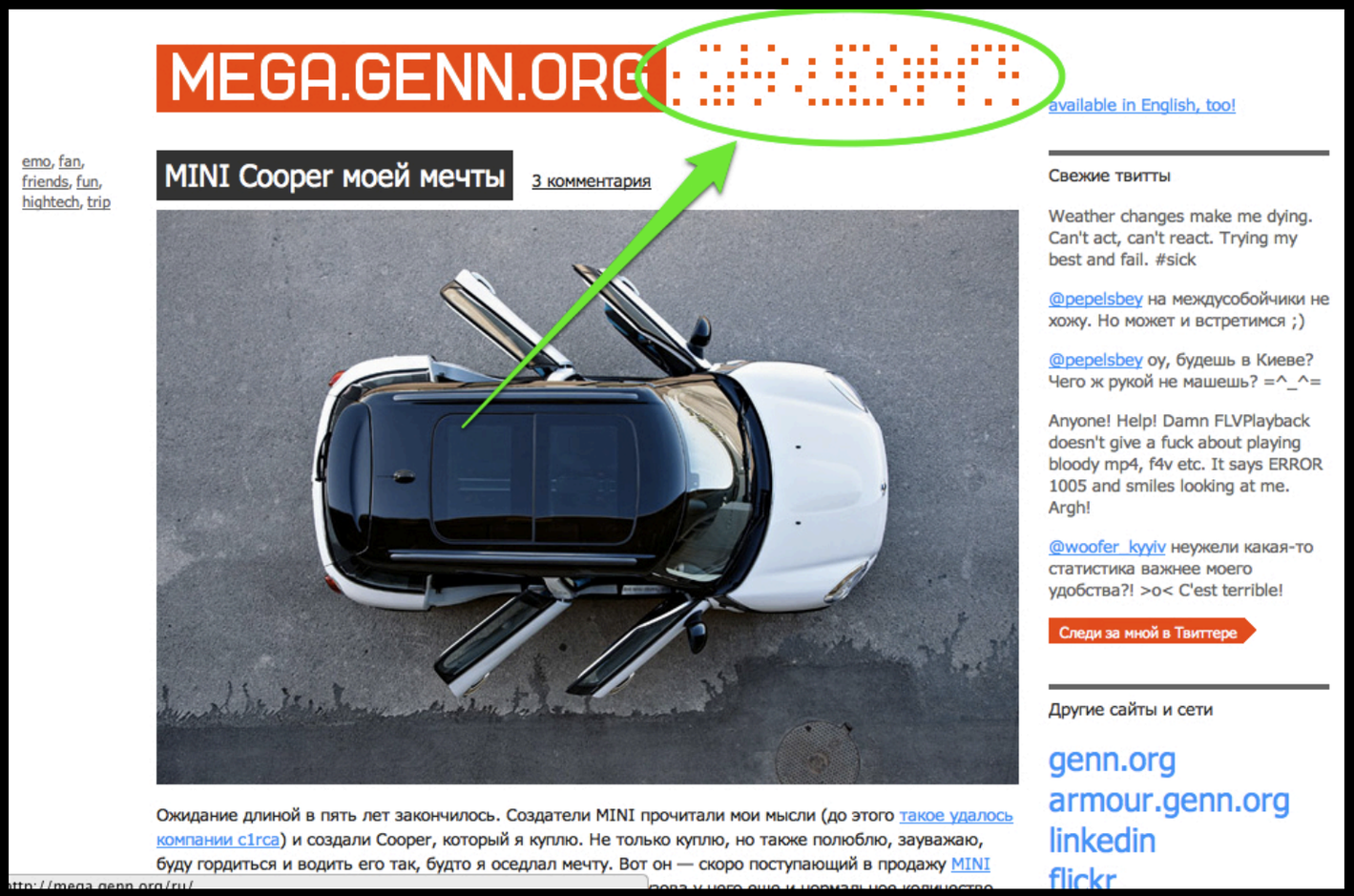
What to do with JS?

- V8 (Google)

What to do with JS?

- V8 (Google)
- SpiderMonkey/TraceMonkey (Mozilla)

Example



I want that dotted pattern.

Don’t ask me why.

Example

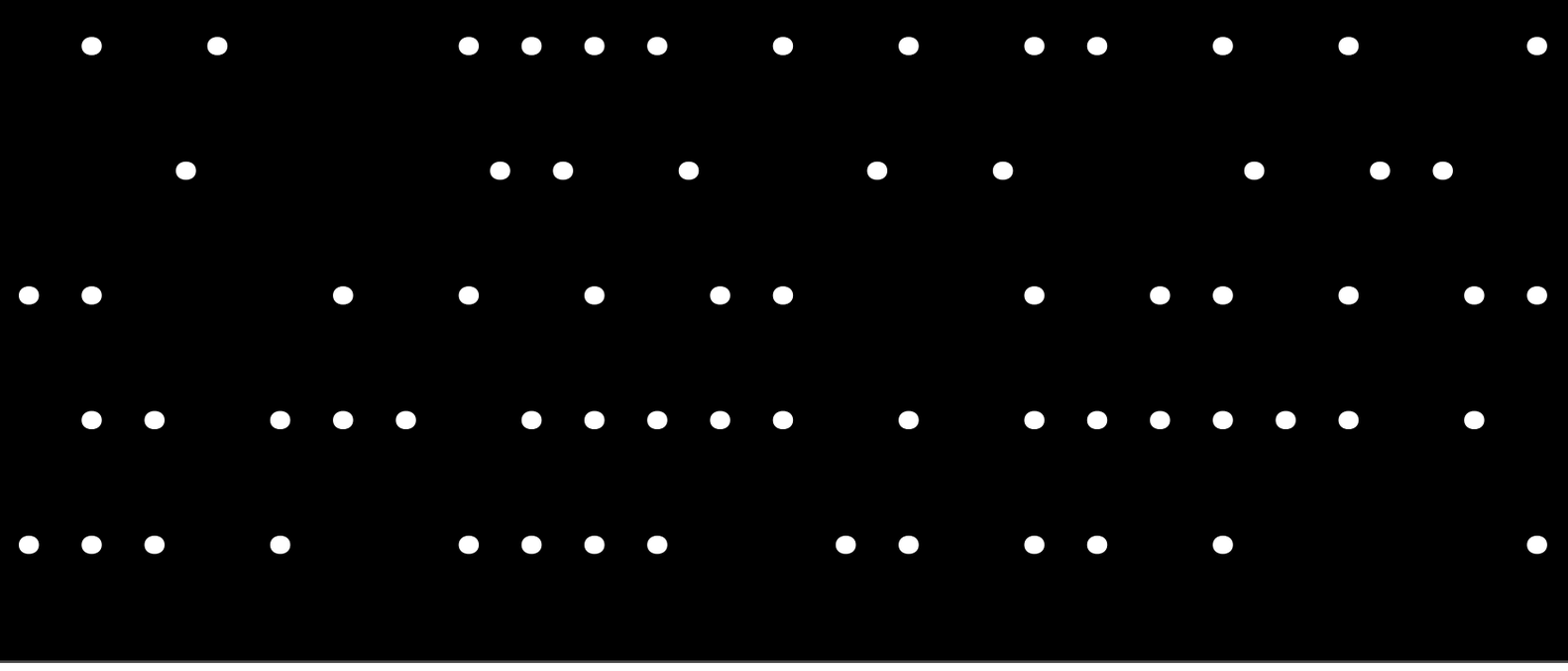
```
require 'harmony'

page = Harmony::Page.fetch('http://mega.genn.org/2010/ipad/')
page.load('http://code.jquery.com/jquery-1.4.2.min.js')

dots = page.execute_js("$('.mclrs').html()")
```

```
dots = asciify(dots)
```

Example

```
require 'harmony'

page = Harmony::Page.fetch('http://mega.genn.org/2010/ipad/')
page.load('http://code.jquery.com/jquery-1.4.2.min.js')

dots = page.execute_js("$('.mclrs').html()")
```

```
dots = asciify(dots)
```



Long time execution

Long time execution

delayed_job

Long time execution

delayed_job

Simple worker

```
class DelayedParser
  def perform()
    Parser.new.extract_posts
  end
end
```

Long time execution

delayed_job

Simple worker

```
class DelayedParser
  def perform()
    Parser.new.extract_posts
  end
end
```

Simple rake task to run it

```
namespace :parser do
  desc "Parse mega.genn.org using delayed_job"
  task :dj => :environment do
    Delayed::Job.enqueue(DelayedParser.new)
  end
end
```

Long time execution

delayed_job

But sometimes it takes **too long**

Long time execution

loops

Doing small tasks.

Long time execution

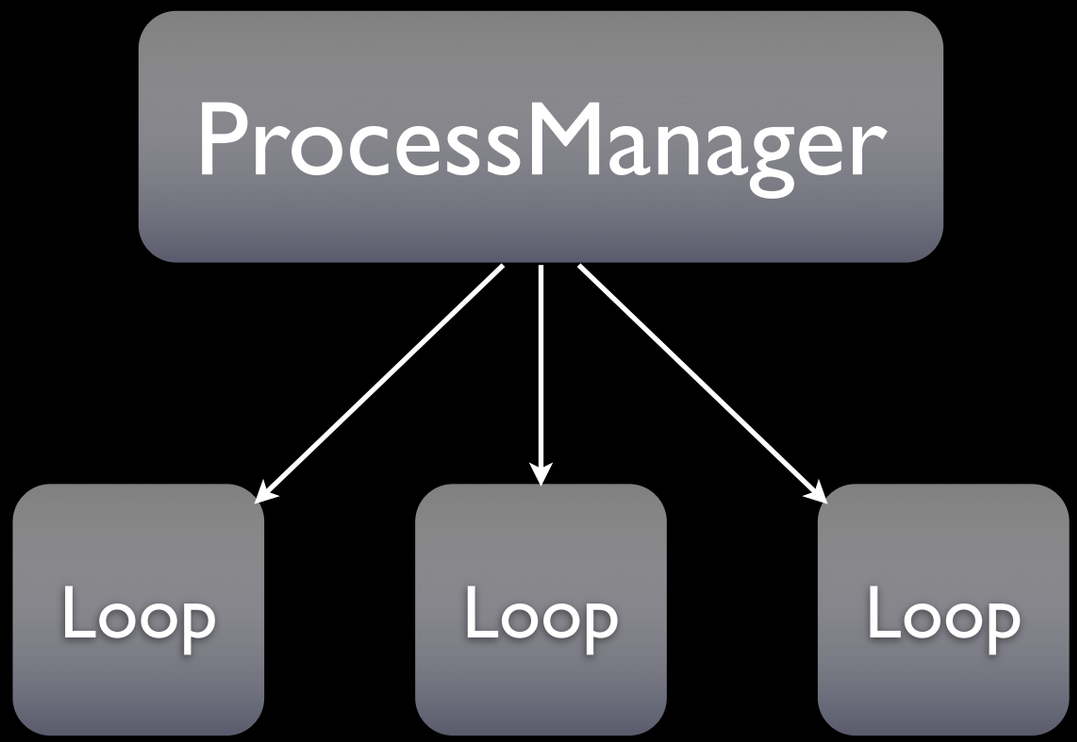
loops

+

RabbitMQ

Doing small tasks.
But at the same time.

Loops



Loops

PostFinderLoop

PostExtractorLoop

RabbitMQ

Messages

routing key = usa.news

routing key = usa.weather

routing key = europe.news

routing key = europe.weather

Broker

Exchange:

Bindings:

Queues:

binding key = usa.#

binding key = #.news

binding key = #.weather

binding key = europe.#

RabbitMQ

genn.pages

genn.posts

http://mega.genn.org/
http://mega.genn.org/page/2/
http://mega.genn.org/page/3/
...

http://mega.genn.org/2010/mini-cooper-i-will-buy/
http://mega.genn.org/2010/ipad/
http://mega.genn.org/2010/prawn-the-queen-salad/
...

RabbitMQ

PostFinderLoop

```
class PostFinderLoop < Loops::AMQP::Bunny
  def process_message(page_url)
    info "Received page url: #{page_url}"
    @parser ||= LoopsParser.new

    @parser.find_posts(page_url) do |post_url|
      @exchange.publish post_url,
        :key => 'genn.posts',
        :persistent => true
    end
  end
end
```

RabbitMQ

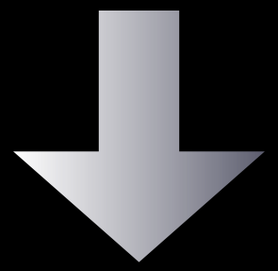
PostExtractorLoop

```
class PostExtractorLoop < Loops::AMQP::Bunny
  def process_message(post_url)
    info "Received post url: #{post_url}"

    @parser ||= LoopsParser.new

    begin
      @parser.extract_post(post_url)
    rescue StandardError => e
      error "Exception #{e} on page (#{post_url})."
    end
  end
end
```

How to test it?

I have no idea.

Links, slides and other



http://github.com/daemon/ruby_barcamp_kiev_03_2010


"That's all Folks!"