

Caching in a multi- language environment

Benjamin Krause <benjamin@omdb.org>



Caching in Rails



Caching in Rails

- ▶ Page Caching
- ▶ Action Caching
- ▶ Fragment Caching



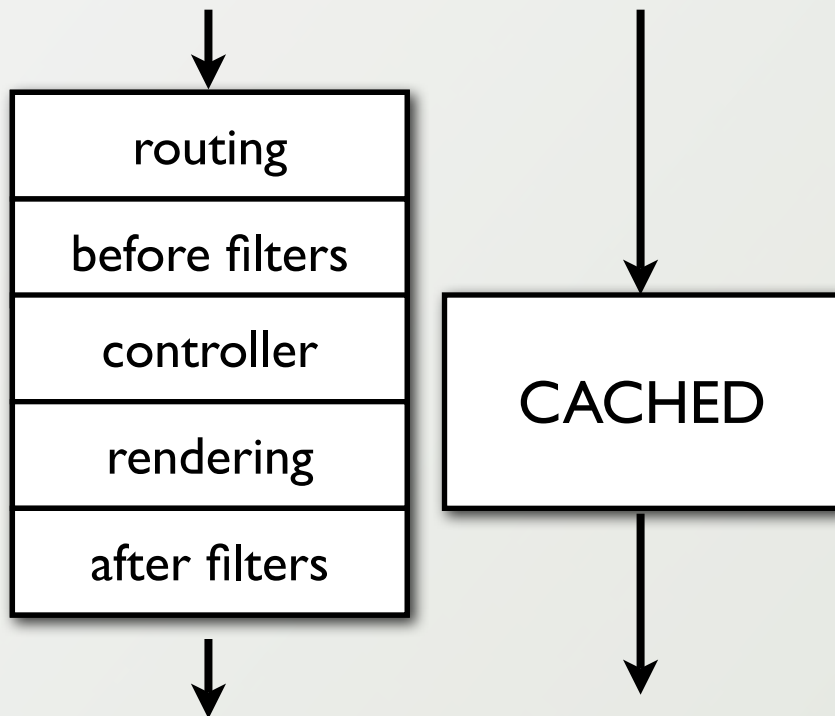
Caching in Rails

- ▶ Page Caching
- ▶ ~~Action Caching~~
- ▶ Fragment Caching



Caching in Rails

Page Caching



Fragment Caching

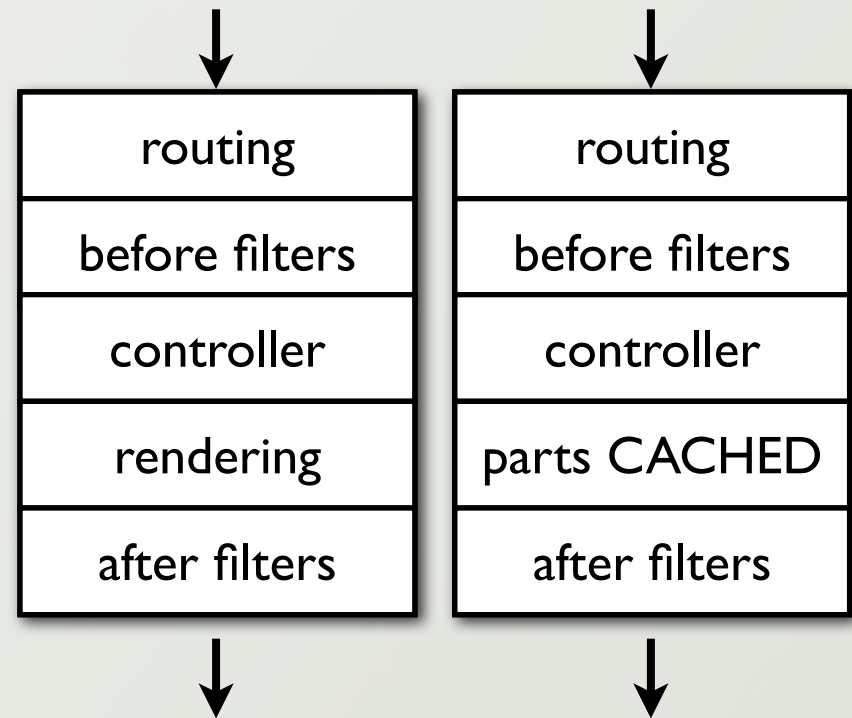
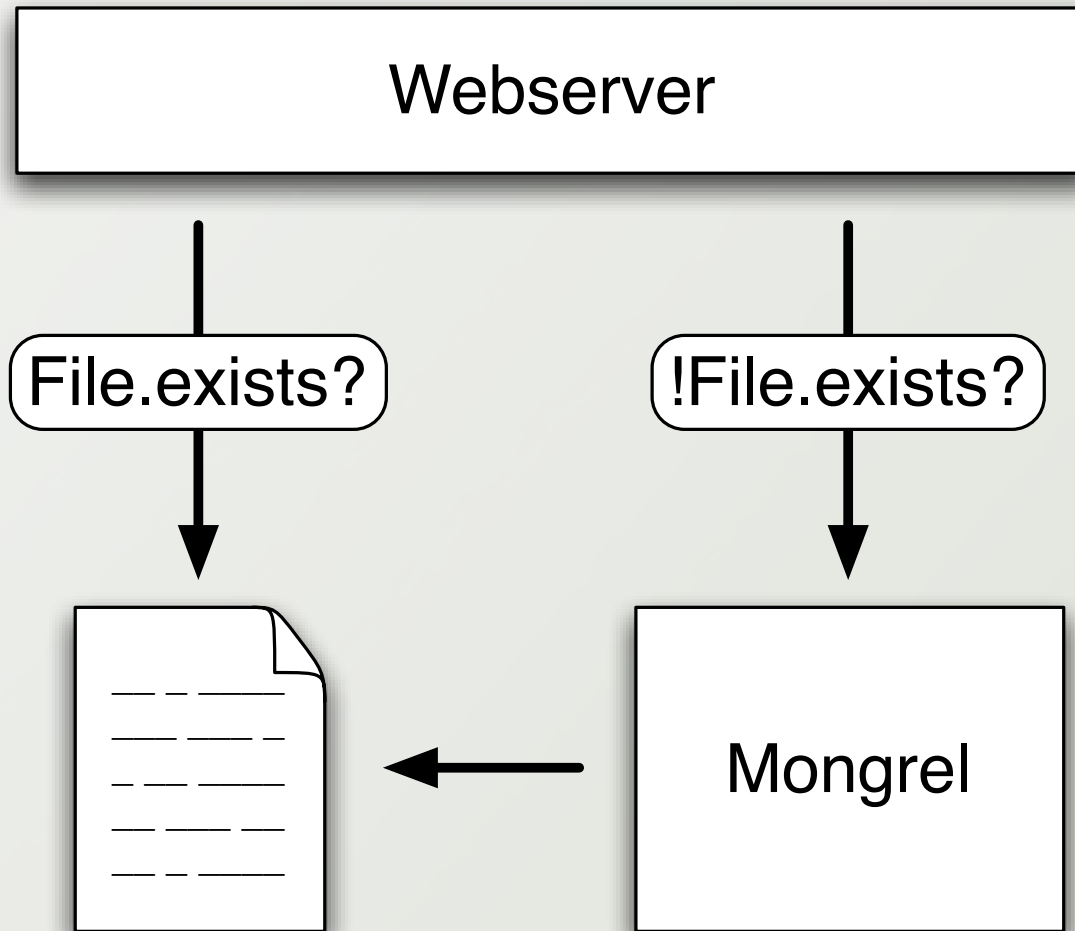


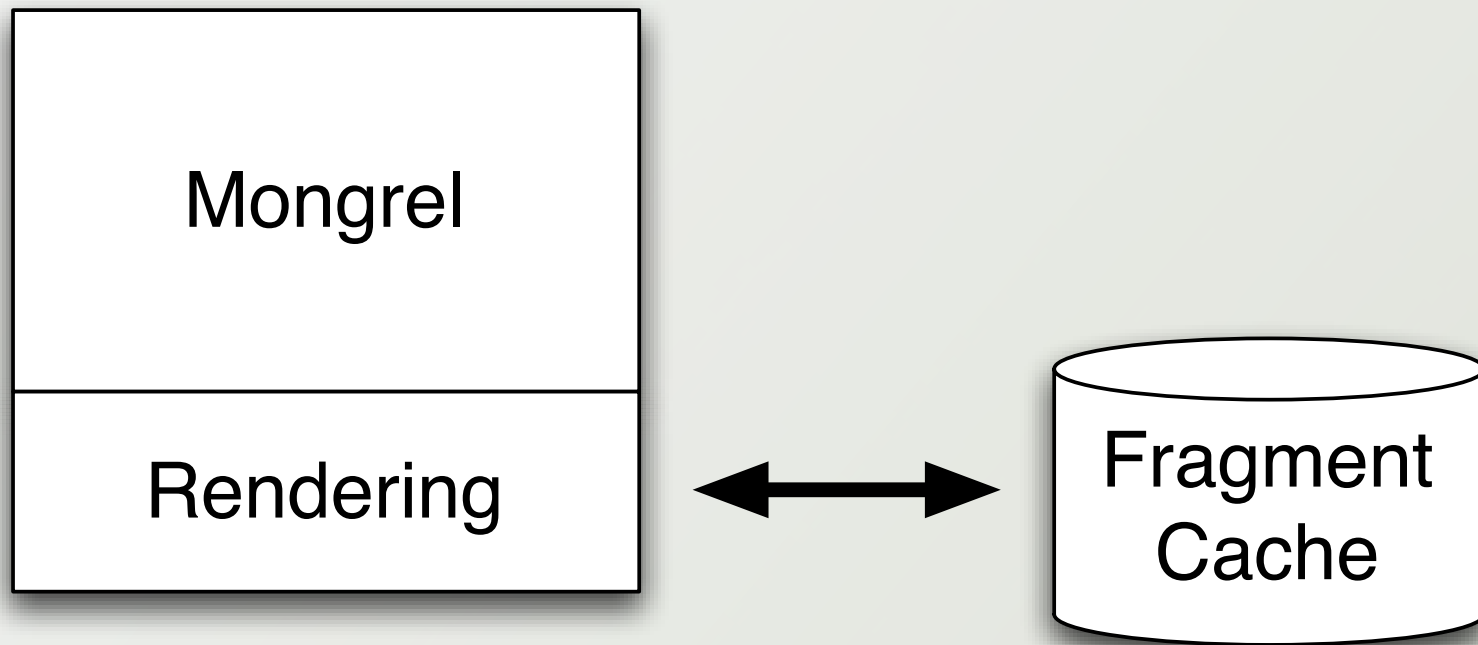
illustration by Tammo Freese



Page Caching



Fragment Caching



Caching in Rails

- ▶ No support for caching in different languages
- ▶ No support to expire all language-caches at once
- ▶ No support for language-negotiation



Content Negotiation

let the user decide, what he wants to see



Content Negotiation

consists of format negotiation and language negotiation



a simple resource

<http://www.omdb.org/movies/2062>



Format Negotiation

- > HTTP “Accept” Header

```
Accept: text/xml,application/xml,  
        application/xhtml+xml,text/html
```

- > Rails supports format negotiation

- > Rails allows you to request a format explicitly (extra routes)



Format Negotiation

<http://www.omdb.org/movies/2062.html>

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Ratatouille</title>
    <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8" />
```

<http://www.omdb.org/movies/2062.xml>

```
<?xml version="1.0" encoding="UTF-8"?>
<movie>
  <name>Ratatouille</name>
  <abstract>
    Remy is a rat, with a talent and passion for cooking.
    He dreams of one day working in a real restaurant
```



Format Negotiation

```
class MoviesController < ApplicationController
  def show
    respond_to do |format|
      format.html { render }
      format.xml  { render :xml => @movie.to_xml }
      format.js   { render :json => @movie.to_json }
    end
  end
end
```



Language Negotiation

- > HTTP “Accept-Language” Header

Accept-Language: en-us,en;q=0.7,de;q=0.3

- > Rails ignores the HTTP Header

- > Rails doesn't add language routes



Language Negotiation

Web pages are sometimes offered in more than one language. Choose languages for displaying these web pages, in order of preference.

Languages in order of preference:

English/United States [en-us]	Move Up
English [en]	Move Down
German [de]	Remove

Japanese [ja]

?

Cancel OK



Language Negotiation

<http://www.omdb.org/movies/2062.html.en>

Routing Error

```
no route found to match "/movies/2062.html.en" with  
{:method=>:get}
```

<http://www.omdb.org/movies/2062.xml.de>

Routing Error

```
no route found to match "/movies/2062.xml.de" with  
{:method=>:get}
```



let's fix that

extending Rails



Language Routes

extending Rails to set language routes



Language Routes

```
# Adding default routes for  
# GET /movies/<id>  
# GET /movies/<id>.<format>  
# PUT /movies/<id>  
# PUT /movies/<id>.<format>  
# ...
```

```
map.resources :movies  
map.resources :people  
map.resources :casts
```



Language Routes

There are several possible routes, how to add a language parameter.

<http://www.omdb.org/movies/2062.html?lang=de>

<http://de.omdb.org/movies/2062.html>

<http://www.omdb.org/de/movies/2062.html>

<http://www.omdb.org/movies/2062.de.html>

<http://www.omdb.org/movies/2062.html.de>



Language Routes

There are several possible routes, how to add a language parameter.

<http://www.omdb.org/movies/2062.html?lang=de>

<http://de.omdb.org/movies/2062.html>

<http://www.omdb.org/de/movies/2062.html>

<http://www.omdb.org/movies/2062.de.html>

<http://www.omdb.org/movies/2062.html.de>



Language Routes

```
# Taken from Rails 1.2.3
# /actionpack-1.13.3/lib/action_controller/resources.rb
# with slight modifications to fit this slide :-)

action_options = action_options_for("show", resource)

map.named_route(
  "#{resource.name_prefix}#{resource.singular}",
  resource.member_path, action_options
)

map.named_route(
  "f_#{@resource.name_prefix}#{@resource.singular}",
  "#{resource.member_path}.:format", action_options
)
```



Language Routes

```
# Taken from Rails 1.2.3
# /actionpack-1.13.3/lib/action_controller/resources.rb
# with slight modifications to fit this slide :-)

action_options = action_options_for("show" resource)
map
  { :conditions => { :method => :get },
    :requirements => { :id => /^[^\/;.,?]+/ },
    :action => "show" }
  resource.member_path, action_options
)

map.named_route(
  "f_#{resource.name_prefix}#{resource.singular}",
  "#{resource.member_path}.:format", action_options
)
```



Language Routes

```
# Taken from Rails 1.2.3
# /actionpack-1.13.3/lib/action_controller/resources.rb
# with slight modifications to fit this slide :-)
```

```
action_options = action_options_for("show" resource)
map
  { :conditions => { :method => :get },
    :requirements => { :id => /^[^\/;.,?]+/ },
    :action => "show" }
  resource.member_path, action_options
)

map.named_route(
  "f_#{resource.name_prefix}#{resource.singular}",
  "#{resource.member_path}.:format", action_options
)
  /movies/:id
```



Language Routes

```
# Extending
# ActionController::Resources::map_new_actions
# ActionController::Resources::map_member_actions
# and others to support a language parameter

map.named_route(
  "fl_#{resource.name_prefix}#{resource.singular}",
  "#{resource.member_path}.:format.:lang",
  action_options
)
```



Language Routes

```
# Extending
# ActionController::Resources::map_new_actions
# ActionController::Resources::map_member_actions
# and others to support a language parameter
```

```
map.named_route(
  "fl_#{resource.name_prefix}#{resource.singular}",
  "#{resource.member_path}.:format.:lang",
  action_options
```

```
) { :conditions => { :method => :get },
  :requirements => { :id => /^[^\/;.,?]+/
                    :lang => /^fr|de|en$/ },
  :action => "show" }
```



Language Routes

<http://www.omdb.org/movies/2062.html.en>

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Ratatouille</title>
    <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8" />
```

<http://www.omdb.org/movies/2062.xml.de>

```
<?xml version="1.0" encoding="UTF-8"?>
<movie>
  <name>Ratatouille</name>
  <abstract>
    Remy is a rat, with a talent and passion for cooking.
    He dreams of one day working in a real restaurant
```



Accept-Language

extending Rails to know about the requested
language



Accept-Language

```
# in environment.rb
#
# These are custom extensions, this is not part
# of Rails

AC = ActionController
AC::AbstractRequest.default_language = :en
AC::AbstractRequest.acceptable_languages = :fr, :de, :en
```



Accept-Language

```
# Taken from Rails 1.2.3
# /actionpack-1.13.3/lib/action_controller/request.rb

# Returns the accepted MIME type for the request
def accepts
  @accepts ||=
    if @env[ 'HTTP_ACCEPT' ].to_s.strip.empty?
      [ content_type, Mime::ALL ]
    else
      Mime::Type.parse( @env[ 'HTTP_ACCEPT' ] )
    end
end
```



Accept-Language

```
class ActionController::AbstractRequest
  def accepts_languages
    begin
      @accepts_languages ||=
        @env[ 'HTTP_ACCEPT_LANGUAGE' ].split(",").collect { |l|
          l.gsub(/;.*$/, '').gsub(/-.*$/, '').downcase.to_sym
        }.push(default_language).uniq
      @accepts_languages.delete_if { |l|
        !acceptable_languages.include?(l)
      }
    rescue NoMethodError
      @accepts_languages =
        default_language.is_a?(Array) ?
          default_language : [ default_language ]
    end
  end
end
```



Accept-Language

Accept-Language: fr
=> [:fr, :en]

Accept-Language: fr; q=1.0, en; q=0.5, fi; q=0.2
=> [:fr, :en]

Accept-Language: en-us,en;q=0.7,de;q=0.3
=> [:en, :de]

Accept-Language: fi, es
=> [:en]



Accept-Language

```
class ActionController::AbstractRequest
  def language
    params[:lang] || accepts_languages.first
  end
end

class ApplicationController
  before_filter :set_language

  def set_language
    @language = session[:lang] || request.language
    # for Globalize:
    #   @language = Locale.set(@language)
    # for Gettext:
    #   params[:lang] = @language unless params[:lang]
  end
end
```



Language Routes

<http://www.omdb.org/movies/2062.html.en>

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Ratatouille</title>
    <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8" />
```

<http://www.omdb.org/movies/2062.xml.de>

```
<?xml version="1.0" encoding="UTF-8"?>
<movie>
  <name>Ratatouille</name>
  <abstract>
    Eine Gourmet-Ratte muss so einiges erleiden,
    um ihre Lust auf exquisite Speisen befriedigen zu können.
```



Caching

view caching, that is

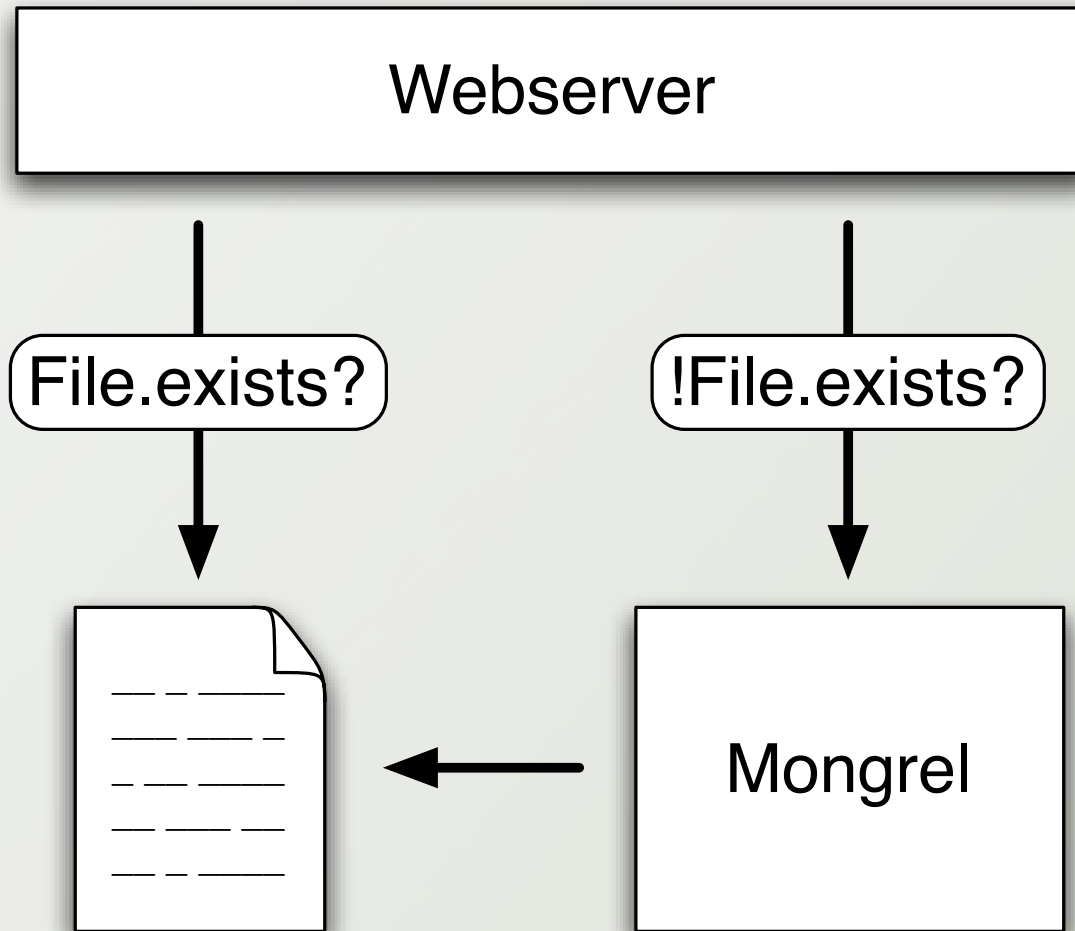


Page Caching

extending Rails to support language-based page
caching



Page Caching



Page Caching

```
class MoviesController < ApplicationController
  caches_page :show

  def show
    respond_to |format| do
      format.html { render }
      format.xml  { render :xml => @movie.to_xml }
      format.js   { render :json => @movie.to_json }
    end
  end
end
```



Page Caching

GET /movies/2062

Mongrel

cache_page

~/public/movies/2062.html



Page Caching

```
# Taken from Rails 1.2.3
# /actionpack-1.13.3/lib/action_controller/caching.rb

def cache_page(content = nil, options = {})
  return unless perform_caching && caching_allowed
  self.class.cache_page(content || response.body,
    url_for(options.merge(:only_path => true,
      :skip_relative_url_root => true,
      :format => params[:format])))
end
```



Page Caching


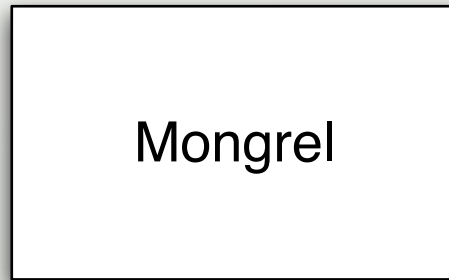
Reimplementing the cache_page method

```
module ActionController::Caching::Pages
  def cache_page(content = nil, options = {})
    return unless perform_caching && caching_allowed
    self.class.cache_page(content || response.body,
      url_for(options.merge(:only_path => true,
        :skip_relative_url_root => true,
        :format => params[:format],
        :lang => request.language)))
  end
end
```




Page Caching

GET /movies/2062



cache_page



A vertical arrow pointing downwards from the "cache_page" box to the file path below.

~/public/movies/2062.html.en



Page Caching

```
curl http://www.omdb.org/movies/2062 \  
-H 'Accept: application/xml'
```

```
=> ~/public/movies/2062.xml.en
```



Page Caching

```
curl http://www.omdb.org/movies/2062 \  
-H 'Accept-Language: fr, en, de'
```

```
=> ~/public/movies/2062.html.fr
```



Page Caching

```
curl http://www.omdb.org/movies/2062 \  
    -H 'Accept-Language: fr, en, de' \  
    -H 'Accept: application/xml'
```

```
=> ~/public/movies/2062.xml.fr
```



Apache

how-to use *Apaches* content negotiations



Apache

Taken from

<http://mongrel.rubyforge.org/docs/apache.html>

Rewrite to check for Rails cached page

RewriteRule ^([\^.]*)\$ \$1.html [QSA]

Redirect all non-static requests to cluster

RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f

RewriteRule ^/(.*)\$ balancer://cluster%{REQUEST_URI}
[P,QSA,L]



Apache

```
# Taken from
# http://bugs.omb.org/browser/trunk/conf/webserver/apache.conf

# Perform content-negotiation and send to mongrel
# if not cached.
RewriteCond %{DOCUMENT_ROOT}/%{LA-U:REQUEST_URI} !-f
RewriteRule ^/(.*)$ balancer://cluster%{REQUEST_URI}
[P,QSA,L]
```



Apache

Apache 2.2 mod_rewrite documentation:

`%{LA-U:variable}` can be used for look-aheads which perform an internal (URL-based) sub-request to determine the final value of variable.



Apache

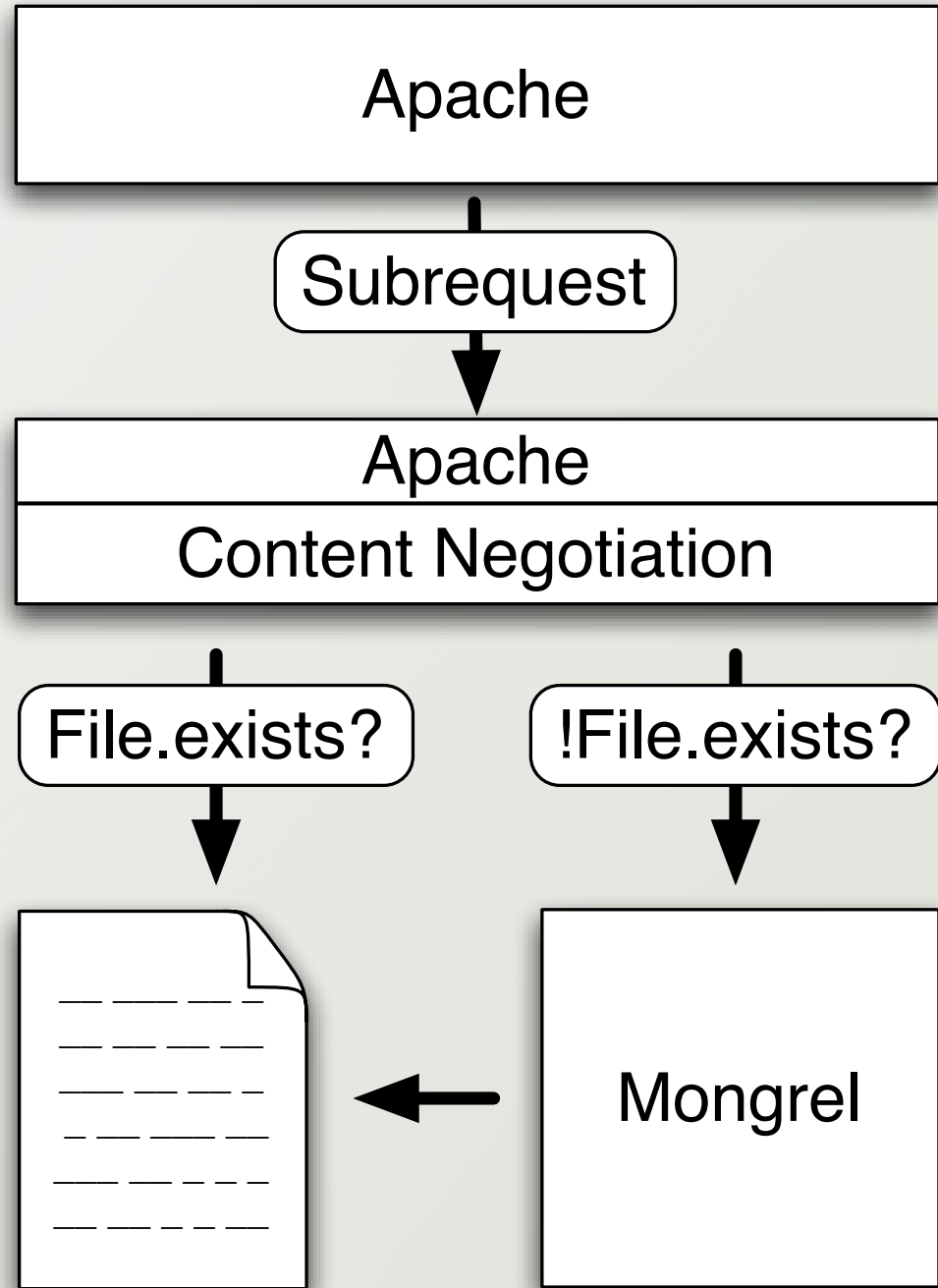
```
# Prefer text/html over other Accept Headers
# Simplified RewriteConditions
RewriteCond %{HTTP_ACCEPT} text/html
RewriteCond %{SCRIPT_FILENAME} !.html$
RewriteCond %{IS_SUBREQ} 'false'
RewriteRule ^/(.*)$ /%{SCRIPT_FILENAME}.html [QSA]

# Perform content-negotiation and send to mongrel
# if not cached.
RewriteCond %{DOCUMENT_ROOT}/%{LA-U:REQUEST_URI} !-f
RewriteCond %{IS_SUBREQ} 'false'
RewriteRule ^/(.*)$ balancer://cluster%{REQUEST_URI}
[P,QSA,L]
```



GET /movies/2062

/movies/2062.html.en



Apache

```
# Let the user change the language in your  
# application
```

```
SetEnvIf Cookie "language=(..)" prefer-language=$1
```



Fragment Caching

extending Rails to support language-based
fragment caching



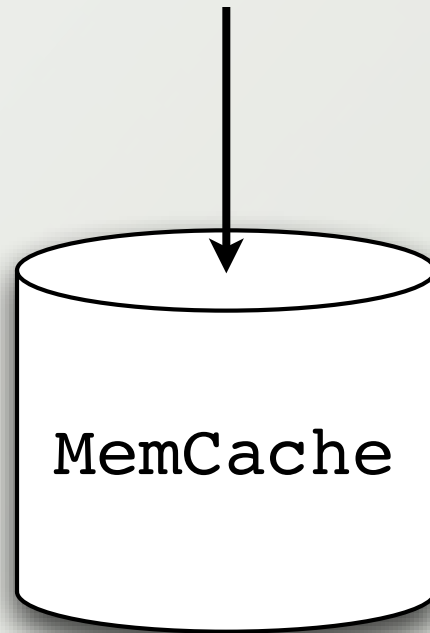
Fragment Caching

```
<% cache do %>
  <div class="person">
    <div class="image">
      <%= img_tag image_url(cast.person.image) %>
    </div>
    <strong>
      <%= link_to cast.person.name,
                  person_url(cast.person) %>
    </strong>
  </div>
<% end %>
```



Fragment Caching

cached content



Fragment Caching

```
<% cache :lang => request.language do %>
  <div class="person">
    <div class="image">
      <%= img_tag image_url(cast.person.image) %>
    </div>
    <strong>
      <%= link_to cast.person.name,
                  person_url(cast.person) %>
    </strong>
  </div>
<% end %>
```



Fragment Caching

```
<% cache :lang => request.language do %>
  <div class="person">
    <div class="image">
      <%= img_tag image_url(cast.person.image) %>
    </div>
    <strong>
      <%= link_to cast.person.name,
                  person_url(cast.person) %>
    </strong>
  </div>
<% end %>
```



Fragment Caching

```
ActionController::Base.fragment_cache_store =  
  :mem_cache_store, "localhost"
```



Fragment Caching

```
ActionController::Base.fragment_cache_store =  
  OMDb::Cache::FragmentCache.instance
```



Fragment Caching

```
ActionController::Base.fragment_cache_store =  
  OMDb::Cache::FragmentCache.instance
```

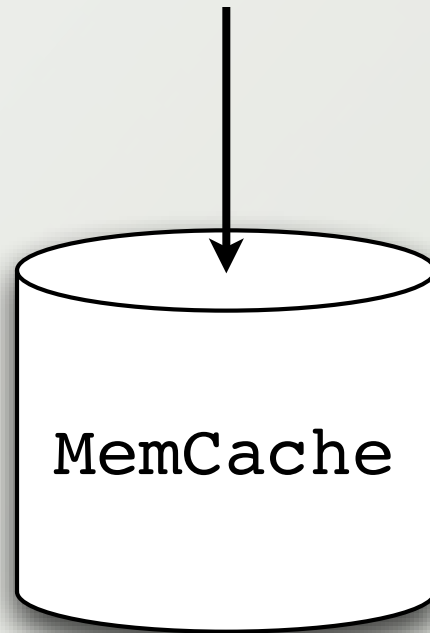
```
# Setting the default MemCache server
```

```
OMDb::Cache::MemCacheStore.servers = 'localhost'
```

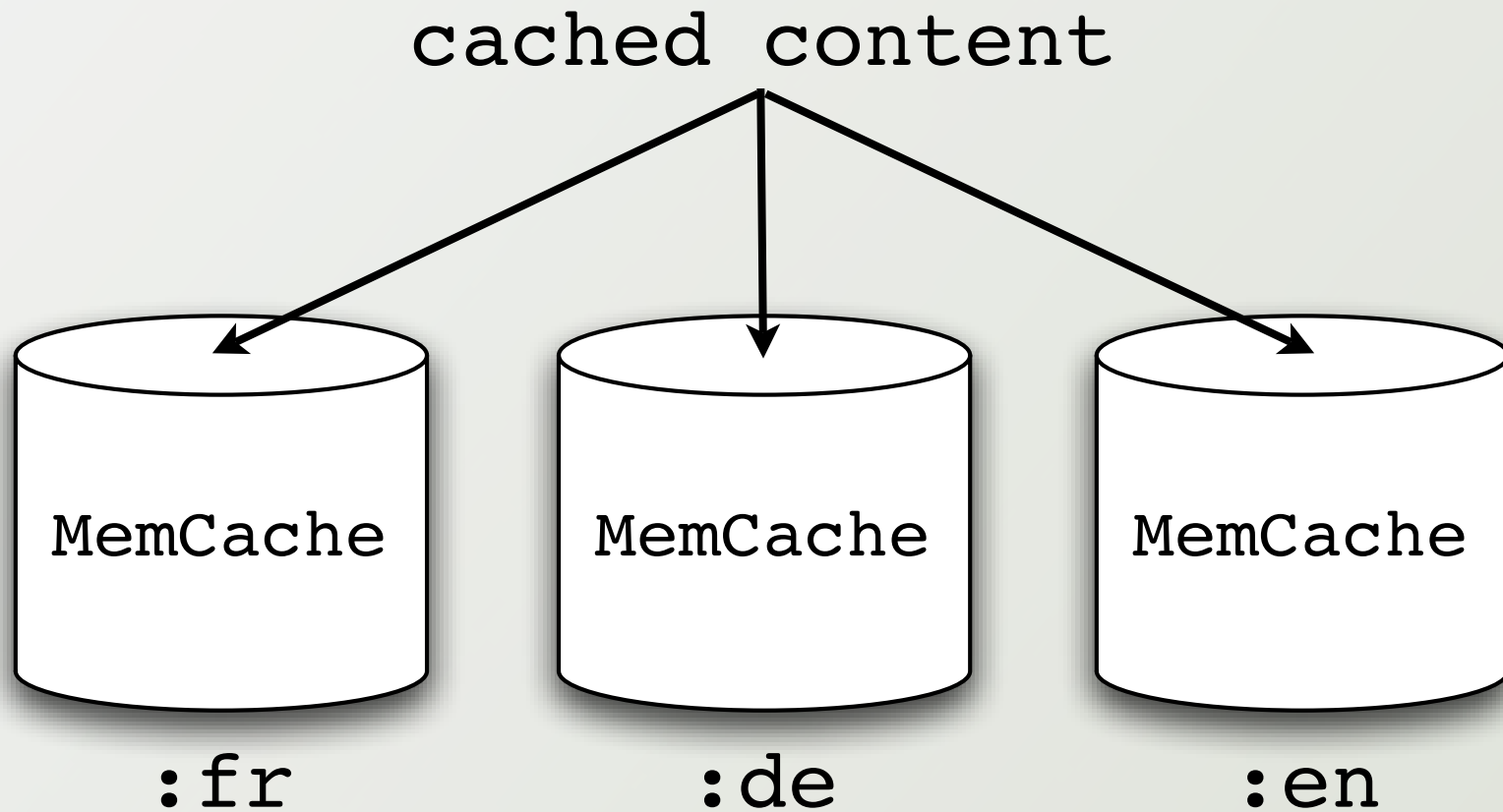


Fragment Caching

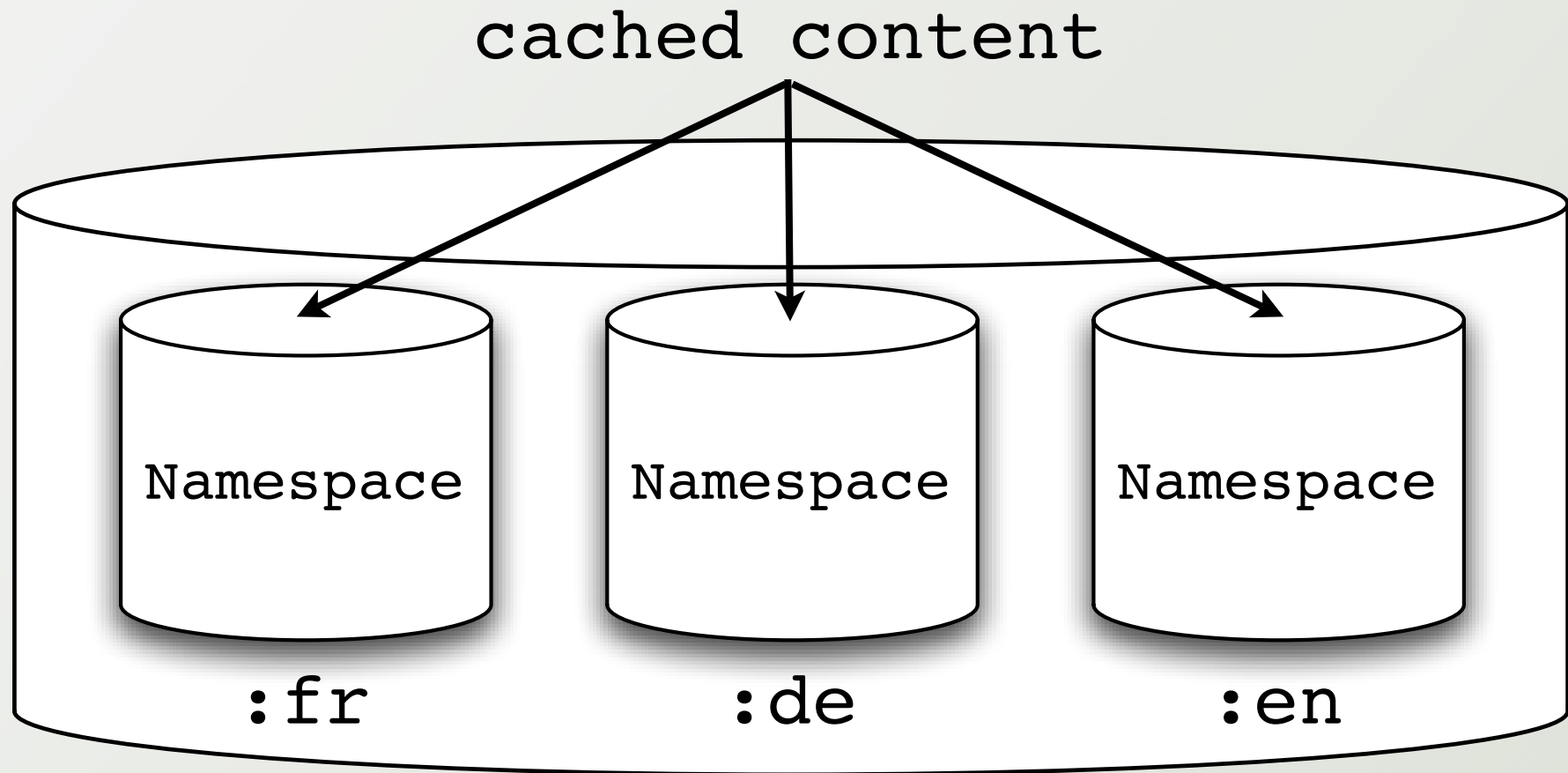
cached content



Fragment Caching

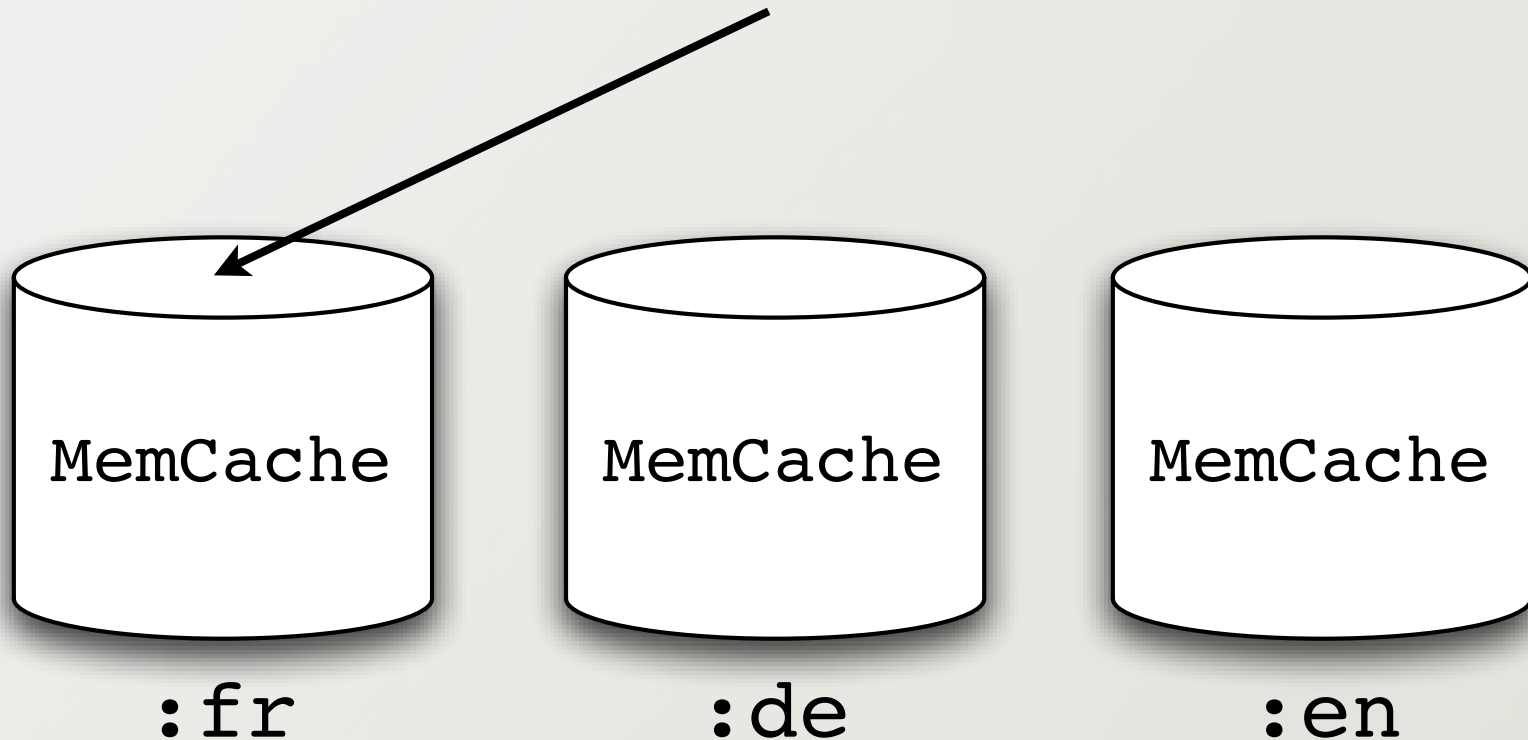


Fragment Caching



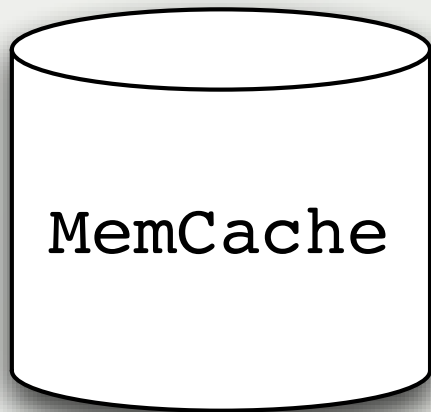
Fragment Caching

<http://www.omdb.org/movies/2062.html.fr>

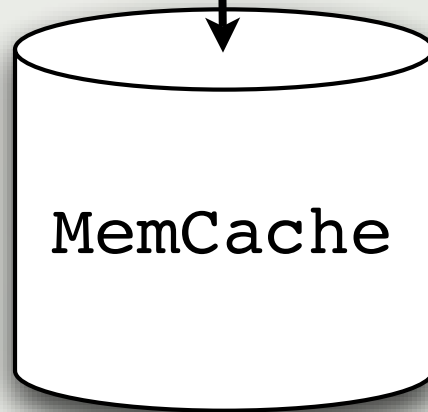


Fragment Caching

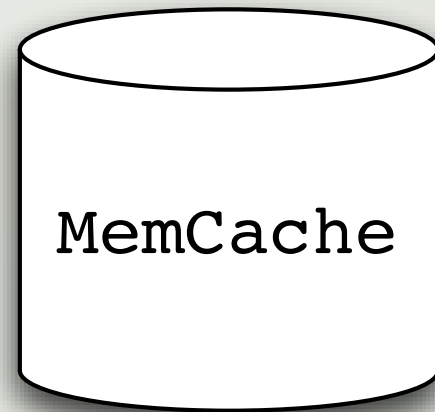
<http://www.omdb.org/movies/2062.html.de>



:fr



:de

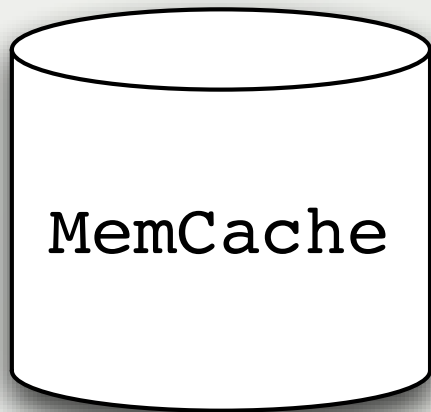


:en

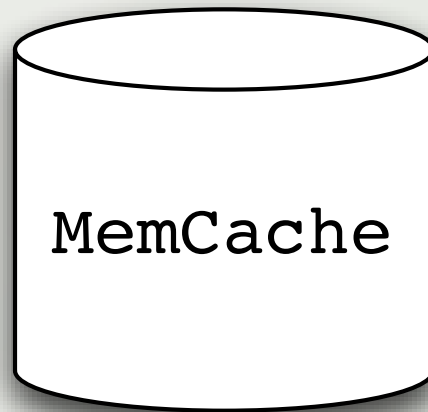


Fragment Caching

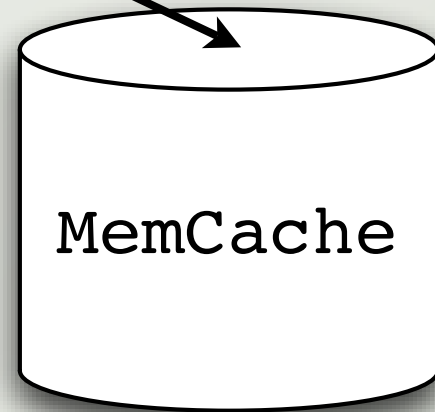
<http://www.omdb.org/movies/2062.html.en>



:fr



:de



:en



Fragment Caching

```
# Delete the fragment for one language
expire_fragment :controller => 'movies',
                 :action    => 'show',
                 :id        => 2062,
                 :lang      => :en
```

```
# delete the fragment for all languages
expire_fragment :controller => 'movies',
                 :action    => 'show',
                 :id        => 2062,
```



Tests

how to test the views



Test Page Caching

```
def test_caching_show
  accept :xml
  # language is a custom extension
  language :de

  page = "/casts/#{casts(:first).id}.xml.de"
  assert_cached_page( page ) do
    get :show, :id => casts(:first).id
    assert_response :success
  end
end
```



Test Page Caching

```
def test_expire_show
  accept :xml
  language :de

  page = "/casts/#{casts(:first).id}.xml.de"
  assert_no_cached_page( page ) do
    put :update, :id => casts(:first).id,
           :job => Job.composer.id
    assert_response :success
  end
end
```



Test Fragment Caching

```
def test_expire_fragment
  accept :js
  language :en
  params = { :controller => 'casts',
            :id          => casts(:first).id,
            :template    => 'cast' }

  assert_cached_fragment( params ) do
    put :update, :id => casts(:first).id,
          :job => Job.composer.id
    assert_template 'cast'
  end

  assert_no_cached_fragment( params ) do
    casts(:first).save
  end
end
```



Where to get the plugins

The plugins are available from

<http://svn.omdb-beta.org/plugins/mlr>

<http://svn.omdb-beta.org/plugins/mlcache>

There will be an official announcement on

<http://blog.omdb-beta.org/>



thank you

