INFINUM

Caching with Ruby on Rails

NENAD NIKOLIC

Caching in a Rails app is a little bit like that one friend you sometimes have around for dinner, but should really have around more often.

CACHE MEANING AND DEFINITION

- Store away in hiding or for future use.
- As developers, we hear the word “cache” quite often. Actually, it means “to hide” in French (“cache-cache” is a hide-and-seek game)

TYPES OF CACHES

- Browser cache
- Memory cache
- Disk cache
- Processor cache
- HTTP cache

WHY CACHING?

- performance
- performance is a feature
- response time in man–computer interaction
- when should you start paying attention to performance

WHEN CACHING CAN'T HELP YOU

- if you have a lot of posts
- realtime apps, chats etc.

HOW ~~RAILS~~ BASECAMP DOES CACHING

THREE TYPES OF CACHING

- page caching
- action caching
- fragment caching (Russian doll caching)

PAGE CACHING

- removed from rails in 4.0
- gem: actionpack-page_caching

PAGE CACHING SETUP AND USAGE

```
config.action_controller.page_cache_directory = "#{Rails.root}/public/cached_pages"

class WeblogController < ActionController::Base
  caches_page :show, :new
end
```

ACTION CACHING

- removed from rails in 4.0
- gem: actionpack-action_caching

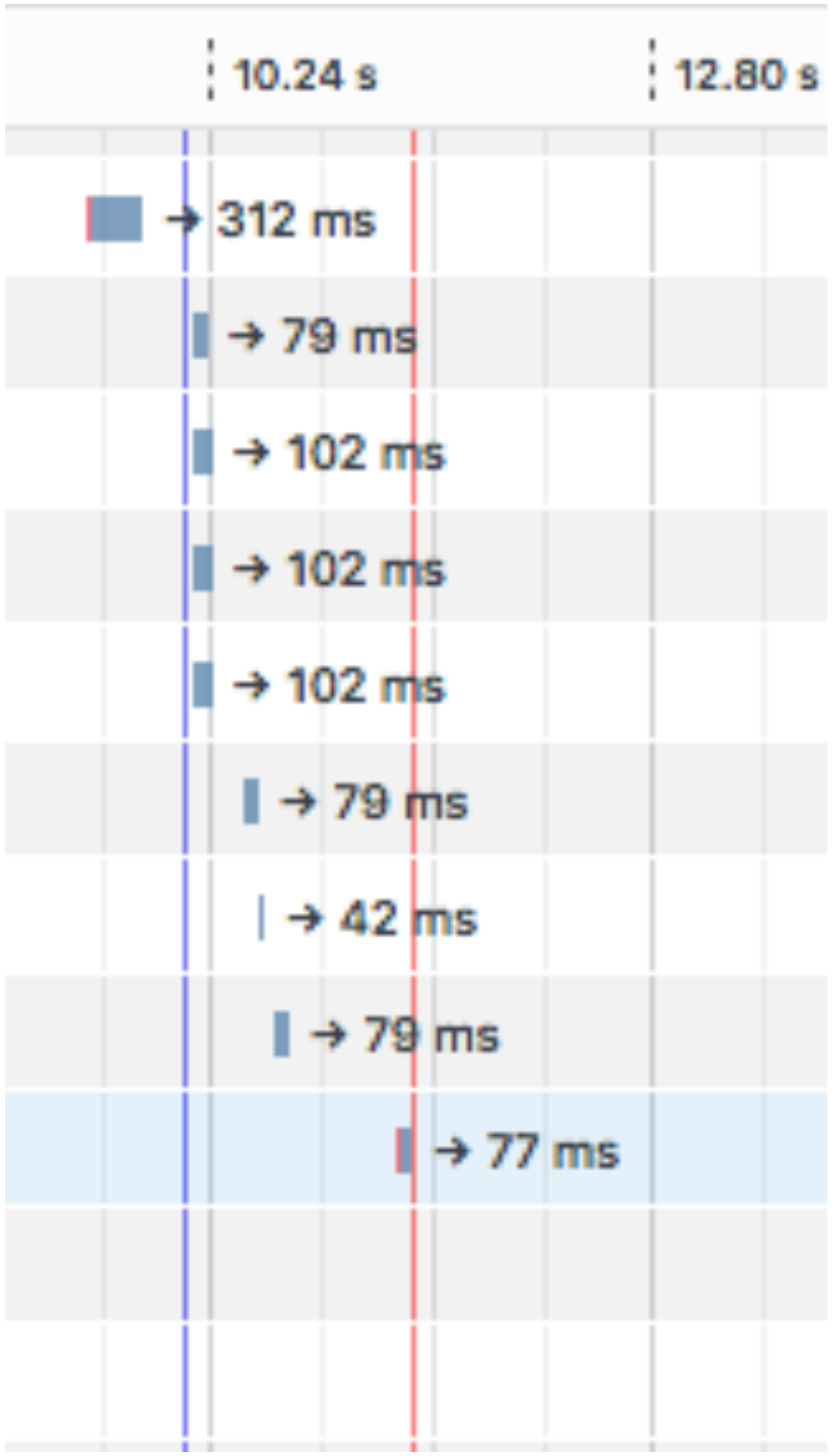
ACTION CACHING USAGE

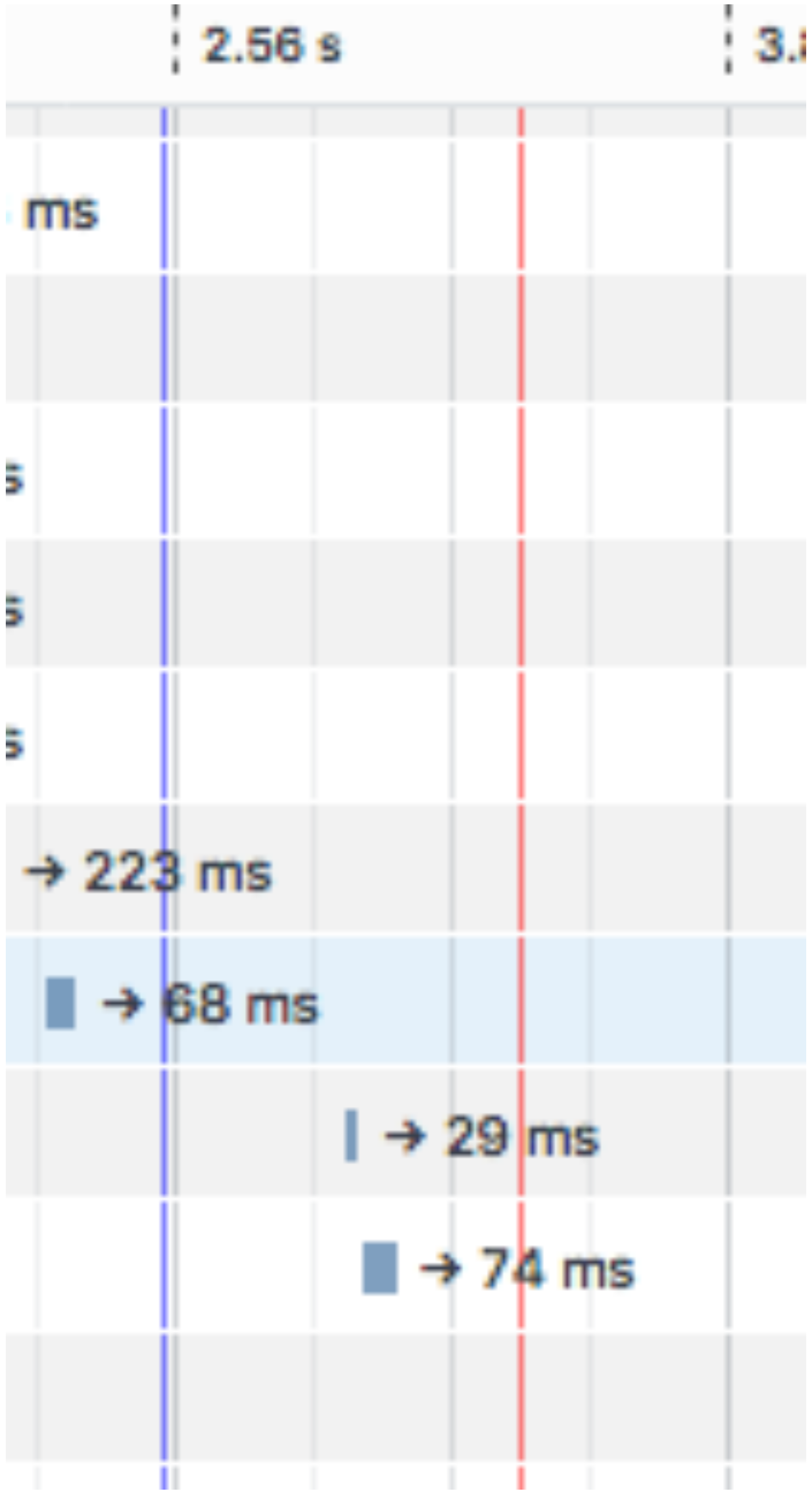
```ruby
class ListsController < ApplicationController
  before_action :authenticate, except: :public

  caches_page   :public
  caches_action :index, :show
end
```

ACTION CACHE USAGE

```ruby
class ListsController < ApplicationController
  # simple fragment cache
  caches_action :current

  # expire cache after an hour
  caches_action :archived, expires_in: 1.hour

  # cache unless it's a JSON request
  caches_action :index, unless: -> { request.format.json? }

  # custom cache path
  caches_action :show, cache_path: { project: 1 }

  # custom cache path with a proc
  caches_action :history, cache_path: -> { request.domain }

  # custom cache path with a symbol
  caches_action :feed, cache_path: :user_cache_path
```

ACTION CACHE MIRAMO EXAMPLE





FRAGMENT CACHING

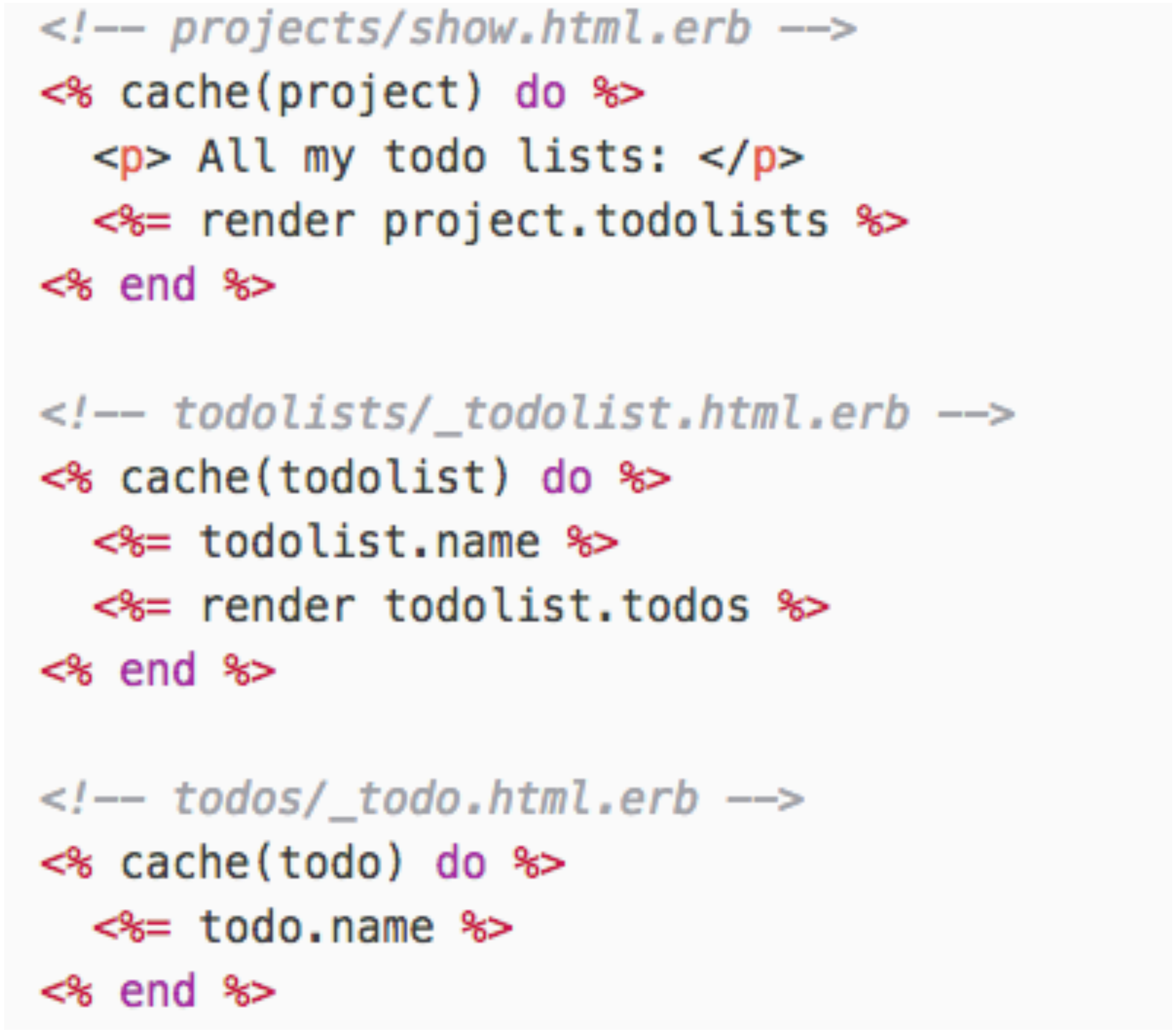
CACHE KEY METHOD

```ruby
Product.new.cache_key
# => "products/new"

Product.find(5).cache_key
# => "products/5" (updated_at not available)

Product.find(5).cache_key
# => "products/5-20071224150000" (updated_at available)
```

CACHE METHOD

```
<% cache project do %>
  <b>All the topics on this project</b>
  <%= render project.topics %>
<% end %>

views/projects/123-20120806214154/7a1156131a6928cb0026877f8b749ac9
      ^class   ^id ^updated_at    ^template tree digest
```

FRAGMENT CACHING EXAMPLE

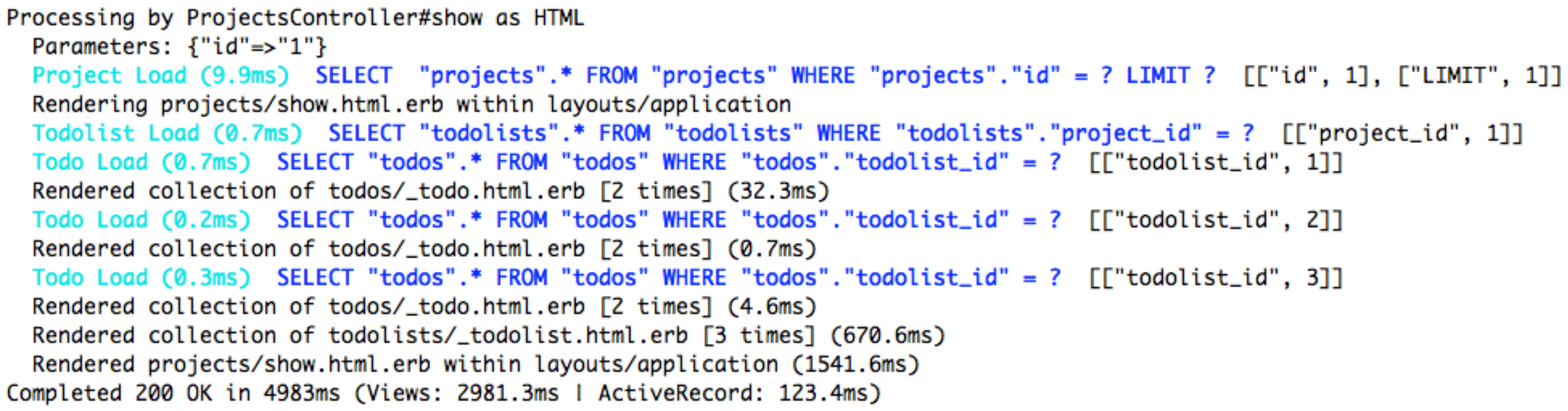


```
<!-- projects/show.html.erb -->
<% cache(project) do %>
  <p> All my todo lists: </p>
  <%= render project.todolists %>
<% end %>

<!-- todolists/_todolist.html.erb -->
<% cache(todolist) do %>
  <%= todolist.name %>
  <%= render todolist.todos %>
<% end %>

<!-- todos/_todo.html.erb -->
<% cache(todo) do %>
  <%= todo.name %>
<% end %>
```

MODELS

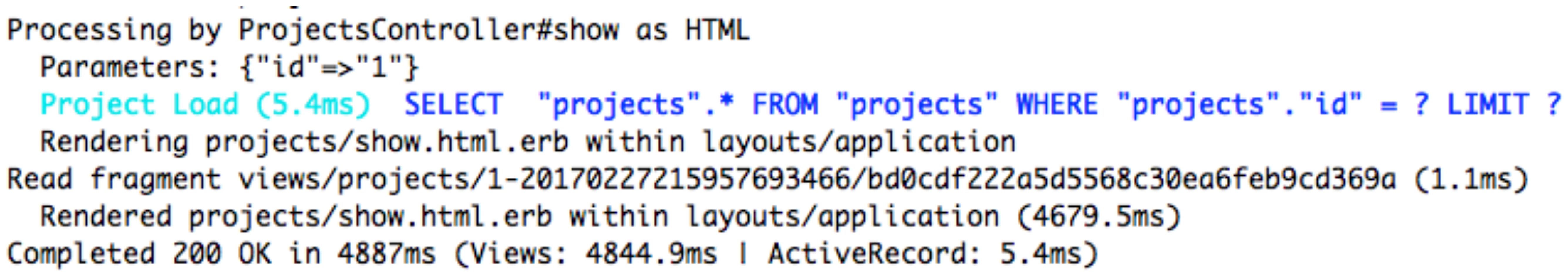
```ruby
class Project < ActiveRecord::Base
end

class Todolist < ActiveRecord::Base
  belongs_to :project, touch: true
end

class Todo < ActiveRecord::Base
  belongs_to :todolist, touch: true
end
```

CASE WITHOUT FRAGMENT CACHING



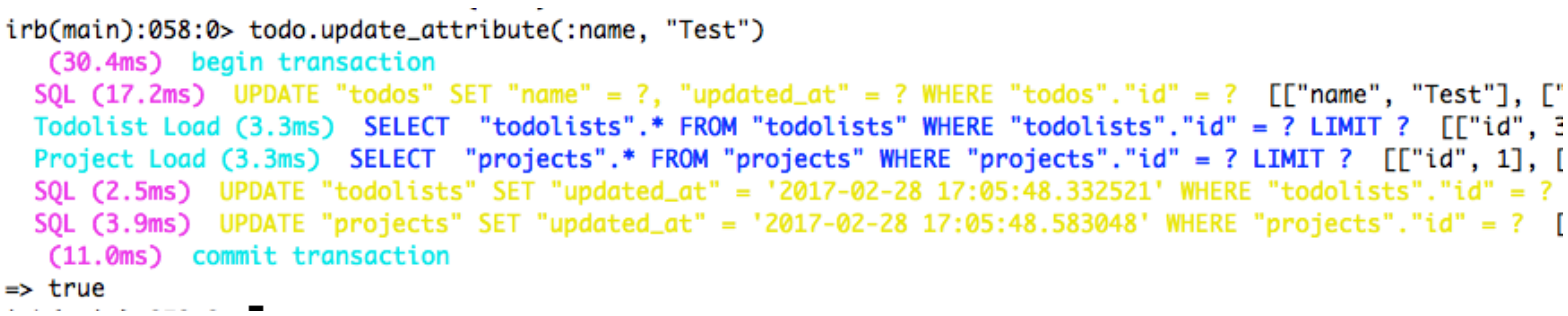
```
Processing by ProjectsController#show as HTML
  Parameters: {"id"=>"1"}
  Project Load (9.9ms)  SELECT  "projects".* FROM "projects" WHERE "projects"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
  Rendering projects/show.html.erb within layouts/application
  Todolist Load (0.7ms)  SELECT "todolists".* FROM "todolists" WHERE "todolists"."project_id" = ?  [["project_id", 1]]
  Todo Load (0.7ms)  SELECT "todos".* FROM "todos" WHERE "todos"."todolist_id" = ?  [["todolist_id", 1]]
  Rendered collection of todos/_todo.html.erb [2 times] (32.3ms)
  Todo Load (0.2ms)  SELECT "todos".* FROM "todos" WHERE "todos"."todolist_id" = ?  [["todolist_id", 2]]
  Rendered collection of todos/_todo.html.erb [2 times] (0.7ms)
  Todo Load (0.3ms)  SELECT "todos".* FROM "todos" WHERE "todos"."todolist_id" = ?  [["todolist_id", 3]]
  Rendered collection of todos/_todo.html.erb [2 times] (4.6ms)
  Rendered collection of todolists/_todolist.html.erb [3 times] (670.6ms)
  Rendered projects/show.html.erb within layouts/application (1541.6ms)
Completed 200 OK in 4983ms (Views: 2981.3ms | ActiveRecord: 123.4ms)
```

WITH CACHING



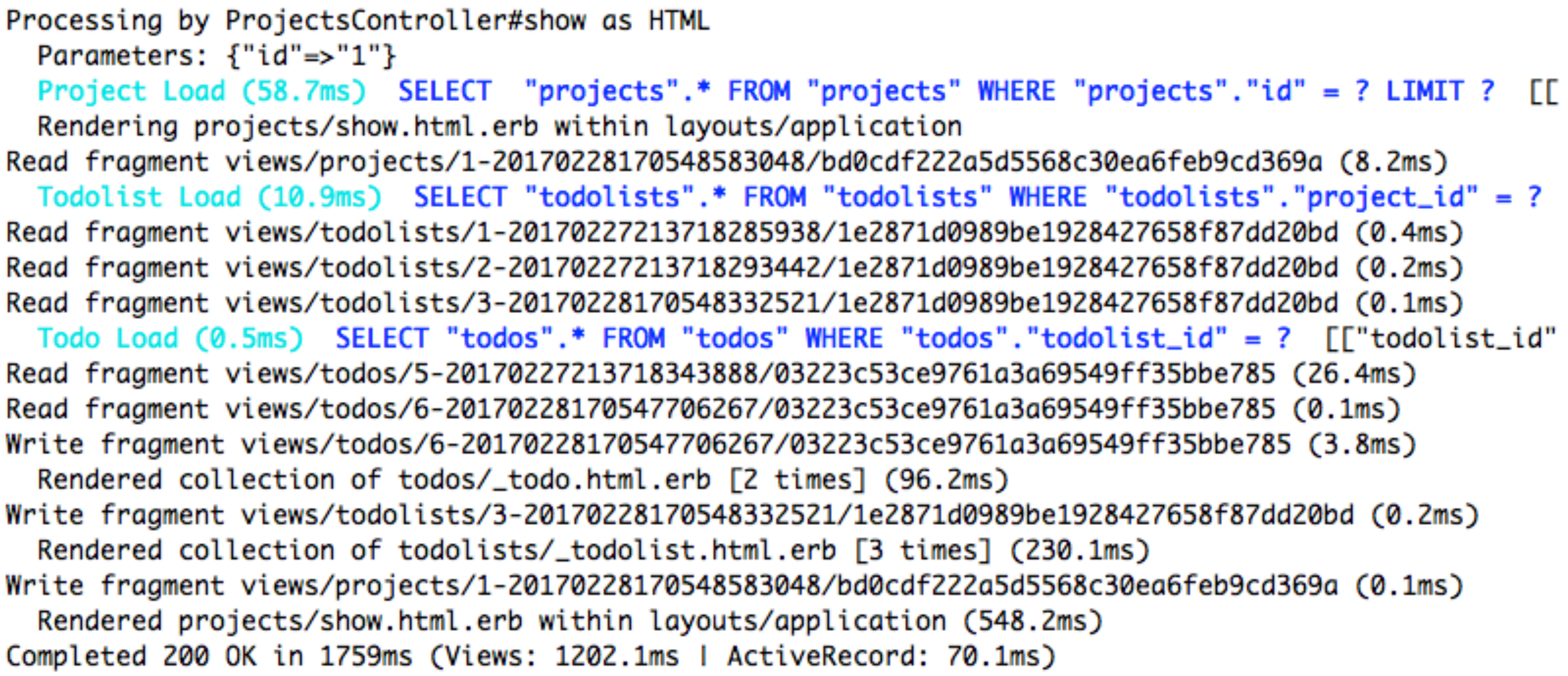
```
Processing by ProjectsController#show as HTML
  Parameters: {"id"=>"1"}
  Project Load (5.4ms)  SELECT  "projects".* FROM "projects" WHERE "projects"."id" = ? LIMIT ?
  Rendering projects/show.html.erb within layouts/application
Read fragment views/projects/1-20170227215957693466/bd0cdf222a5d5568c30ea6feb9cd369a (1.1ms)
  Rendered projects/show.html.erb within layouts/application (4679.5ms)
Completed 200 OK in 4887ms (Views: 4844.9ms | ActiveRecord: 5.4ms)
```

SINGLE TODO UPDATE



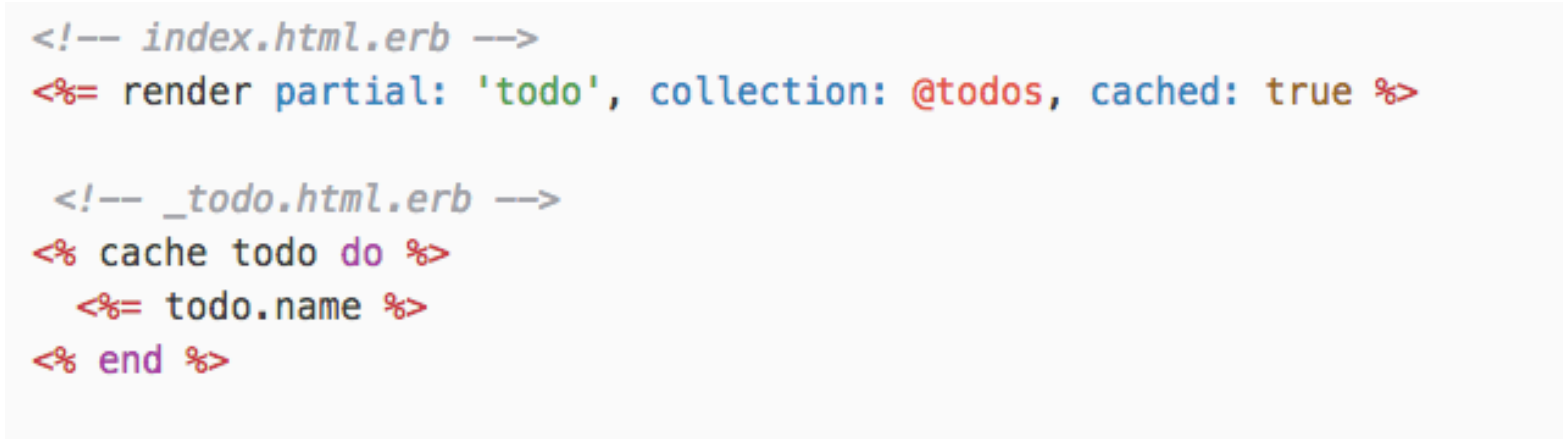
```
irb(main):058:0> todo.update_attribute(:name, "Test")
   (30.4ms)  begin transaction
  SQL (17.2ms)  UPDATE "todos" SET "name" = ?, "updated_at" = ? WHERE "todos"."id" = ?  [["name", "Test"], ["
  Todolist Load (3.3ms)  SELECT  "todolists".* FROM "todolists" WHERE "todolists"."id" = ? LIMIT ?  [["id", 3
  Project Load (3.3ms)  SELECT  "projects".* FROM "projects" WHERE "projects"."id" = ? LIMIT ?  [["id", 1], [
  SQL (2.5ms)  UPDATE "todolists" SET "updated_at" = '2017-02-28 17:05:48.332521' WHERE "todolists"."id" = ?
  SQL (3.9ms)  UPDATE "projects" SET "updated_at" = '2017-02-28 17:05:48.583048' WHERE "projects"."id" = ?  [
   (11.0ms)  commit transaction
=> true
```
]

CACHE AFTER UPDATE



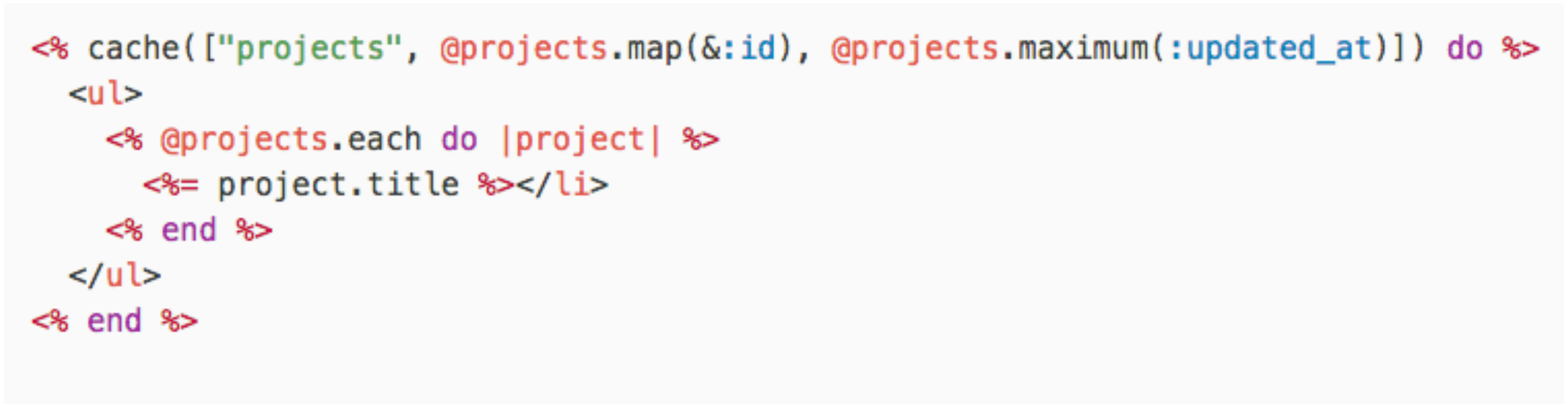
```
Processing by ProjectsController#show as HTML
  Parameters: {"id"=>"1"}
  Project Load (58.7ms)  SELECT  "projects".* FROM "projects" WHERE "projects"."id" = ? LIMIT ?  [[
  Rendering projects/show.html.erb within layouts/application
Read fragment views/projects/1-20170228170548583048/bd0cdf222a5d5568c30ea6feb9cd369a (8.2ms)
  Todolist Load (10.9ms)  SELECT "todolists".* FROM "todolists" WHERE "todolists"."project_id" = ?
Read fragment views/todolists/1-20170227213718285938/1e2871d0989be1928427658f87dd20bd (0.4ms)
Read fragment views/todolists/2-20170227213718293442/1e2871d0989be1928427658f87dd20bd (0.2ms)
Read fragment views/todolists/3-20170228170548332521/1e2871d0989be1928427658f87dd20bd (0.1ms)
  Todo Load (0.5ms)  SELECT "todos".* FROM "todos" WHERE "todos"."todolist_id" = ?  [["todolist_id"
Read fragment views/todos/5-20170227213718343888/03223c53ce9761a3a69549ff35bbe785 (26.4ms)
Read fragment views/todos/6-20170228170547706267/03223c53ce9761a3a69549ff35bbe785 (0.1ms)
Write fragment views/todos/6-20170228170547706267/03223c53ce9761a3a69549ff35bbe785 (3.8ms)
  Rendered collection of todos/_todo.html.erb [2 times] (96.2ms)
Write fragment views/todolists/3-20170228170548332521/1e2871d0989be1928427658f87dd20bd (0.2ms)
  Rendered collection of todolists/_todolist.html.erb [3 times] (230.1ms)
Write fragment views/projects/1-20170228170548583048/bd0cdf222a5d5568c30ea6feb9cd369a (0.1ms)
  Rendered projects/show.html.erb within layouts/application (548.2ms)
Completed 200 OK in 1759ms (Views: 1202.1ms | ActiveRecord: 70.1ms)
```

HANDLING MULTI FETCH



```erb
<!-- index.html.erb -->
<%= render partial: 'todo', collection: @todos, cached: true %>

 <!-- _todo.html.erb -->
<% cache todo do %>
  <%= todo.name %>
<% end %>
```

COLLECTION CACHING



```
<% cache(["projects", @projects.map(&:id), @projects.maximum(:updated_at)]) do %>
  <ul>
    <% @projects.each do |project| %>
      <%= project.title %></li>
    <% end %>
  </ul>
<% end %>
```

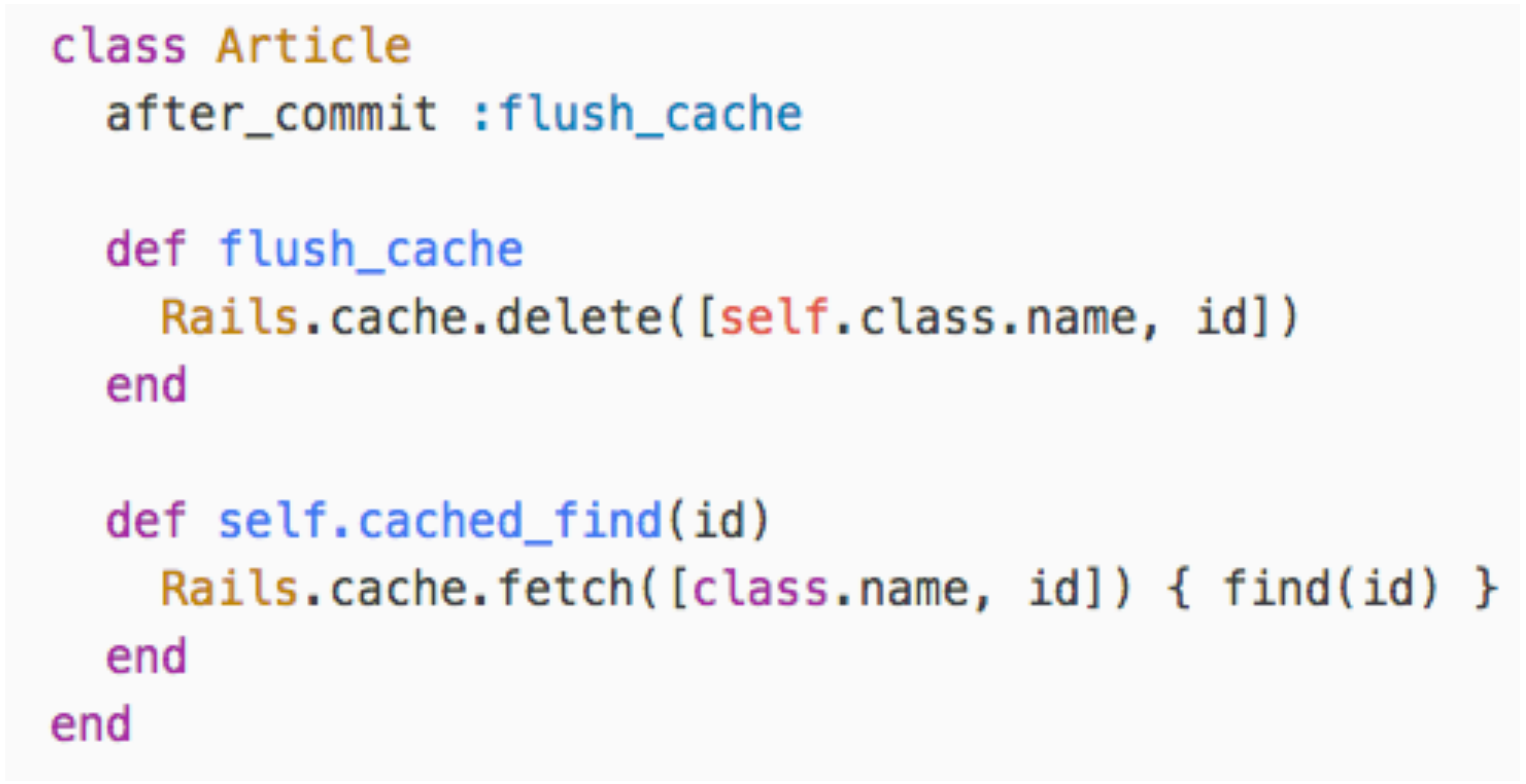
RAILS 5 COLLECTION CACHING

```
@users.cache_key
=> "users/query-67ed32b36805c4b1ec1948b4eef8d58f-3-20160116111659084027"

<% cache(@projects) do %>
  <% @projects.each do |project| %>
    <%= project.title %>
  <% end %>
<% end %>
```

LOW LEVEL CACHING

```ruby
#Basic read write
Rails.cache.write('date', Date.today)
Rails.cache.read('date')

Rails.cache.fetch('date') { Date.today }
Rails.cache.fetch('date', expires_in: 5.minutes) { Date.today }
```

LOW LEVEL CACHING EXAMPLE



```ruby
class Article
  after_commit :flush_cache

  def flush_cache
    Rails.cache.delete([self.class.name, id])
  end

  def self.cached_find(id)
    Rails.cache.fetch([class.name, id]) { find(id) }
  end
end
```

LOW LEVEL CACHING EXAMPLE

```
def Product.out_of_stock
  Rails.cache.fetch("out_of_stock_products", :expires_in => 5.minutes) do
    Product.all.joins(:inventory).conditions.where("inventory.quantity = 0")
  end
end
```

CACHE BACKENDS

ACTIVESUPPORT::FILESTORE

PROS

- FileStore works across processes
- Disk space is cheaper than RAM

CONS

- Filesystems are slow(ish)
- Caches can't be shared across hosts
- Not an LRU cache
- Crashes Heroku dynos

ACTIVESUPPORT::MEMORYSTORE

PROS

- It's fast
- It's easy to set up

CONS

- Caches can't be shared across processes or hosts
- Caches add to your total RAM usage

MEMCACHE AND DALLI

PROS

- Distributed, so all processes and hosts can share

CONS

- Distributed caches are susceptible to network issues and latency
- Expensive
- Cache values are limited to 1MB

REDIS AND REDIS-STORE

PROS

- Distributed, so all processes and hosts can share
- Allows different eviction policies beyond LRU
- Can persist to disk, allowing hot restarts

CONS

- Distributed caches are susceptible to network issues and latency
- Expensive
- While Redis supports several data types, redis-store only supports Strings

CONCLUSION

- page caching
- action caching
- fragment caching (Russian doll caching)
- low hanging fruit
- don’t be afraid to write custom caching techniques