In this lecture, we will discuss…

- Cache Control
- Delegation of Responsibility
- “`expires`”

Cache Control

- *Used to specify directives that must be obeyed by all caching mechanisms along the request-response chain*
- Provide better hints to the client as to how long the information is good

Demo

- Request Movie 10 times (rapid fire)

```
> 10.times.each {HTTParty.head("http://localhost:3000/movies/12345")}
 => 10
> pp MovieAccess.where(:movie_id=>"12345",:action=>"show-stale").pluck(:created_at, :action)
[[2016-01-12 19:26:20 UTC, "show-stale"],
 [2016-01-12 19:26:20 UTC, "show-stale"],
 [2016-01-12 19:26:20 UTC, "show-stale"],
 [2016-01-12 19:26:20 UTC, "show-stale"],
 [2016-01-12 19:26:20 UTC, "show-stale"],
 [2016-01-12 19:26:20 UTC, "show-stale"],
 [2016-01-12 19:26:21 UTC, "show-stale"],
 [2016-01-12 19:26:21 UTC, "show-stale"],
 [2016-01-12 19:26:21 UTC, "show-stale"],
 [2016-01-12 19:26:21 UTC, "show-stale"]]
```

- Each call results in a database access (no headers)

Delegate Responsibility

- Update the show method to include two caching headers:
  - Expires and Cache-Control
  - Overlap in meaning and if they ever conflict, `Cache-Control` is supposed to take precedence

Delegate Responsibility

- Document will expire at a certain time
- Document is not specific to an individual caller
  - You may cache this document for other callers as well
  - If this information was specific to the caller (e.g., a personal bank statement), then `Cache-Control` would either be set to `nocache` or private to keep the resource from being served to other clients
- The maximum time to cache = 10 seconds

expires

- Rails method - set the Cache-Control response header

```
  def show
    @movie.movie_accesses.create(:action=>"show")
    if stale? @movie
      @movie.movie_accesses.create(:action=>"show-stale")
      #do some additional, expensive work here
    secs=10
    response.headers["Expires"] = secs.seconds.from_now.httpdate
#    response.headers["Cache-Control"] = "public, max-age=#{secs}"
    expires_in secs.seconds, :public=>true
    end
```

Sample Call: return message

- Sample response

```
etag:
- '"dd7543eb8124a81a065c2d0629222e2c"'
last-modified:
- Tue, 12 Jan 2016 17:16:35 GMT
expires:
- Tue, 12 Jan 2016 19:52:25 GMT
cache-control:
- max-age=10, public
```

Changes

- Add gems
  - gem 'httparty'
  - gem 'dry_ice‘
- app/services/

```ruby
# app/services/cached_ws.rb
class CachedWS
  include HTTParty
  include HTTParty::DryIce
#  debug_output $stdout
  base_uri "http://localhost:3000"
  cache Rails.cache
end
```

Demo

- ✧ Script – DB is polled every 9 to 12 seconds
  - 3 second sleep and 10 second cache timeout

```
> 10.times.each do |x|
   p "look=#{x}, accesses=#{Movie.find("12345").movie_accesses.where(:action=>"show").count}"
   CachedWS.get("/movies/12345.json").parsed_response
   sleep(3.seconds)
   end
"look=0, accesses=0"
"look=1, accesses=1"
"look=2, accesses=1"
"look=3, accesses=1"
"look=4, accesses=1"
"look=5, accesses=2"
"look=6, accesses=2"
"look=7, accesses=2"
"look=8, accesses=2"
"look=9, accesses=3"
```

Service Side Caching

Demo

Summary

- Cache Control techniques can be done to offload some work that may not have to be done during steady-state

What's Next?

- Server Caching