Stefan Tilkov, innoQ Deutschland GmbH

The State of RESTful Rails

http://railsconsulting.de | http://www.innoq.com

Stefan Tilkov
stefan.tilkov@innoq.com
http://www.innoq.com/blog/st/
@stilkov

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 21 02 77 162-100

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

info@innoq.com · www.innoq.com

http://rest-http.info

http://Heise.de/developer/podcast

REST

The REST Uniform Interface

identification of resources

resource manipulation through representations

self-descriptive messages

hypermedia as the engine of application state

The REST Uniform Interface

identification of resources

resource manipulation through representations

self-descriptive messages

hypermedia as the engine of application state

http://example.com/orders?year=2008

http://example.com/customers/1234

http://example.com/orders/2007/10/776654

http://example.com/products/4554

http://example.com/processes/sal-increase-234

The REST Uniform Interface

identification of resources

resource manipulation through representations

self-descriptive messages

hypermedia as the engine of application state

```
GET /customers/1234
Host: example.com
Accept: application/vnd.mycompany.customer+xml
```

```
<customer>...</customer>
```

```
GET /customers/1234
Host: example.com
Accept: text/x-vcard
```

```
begin:vcard
...
end:vcard
```

The REST Uniform Interface

identification of resources

resource manipulation through representations

self-descriptive messages

hypermedia as the engine of application state

Standard Method

Media Type

Control Data

Data


```
GET /service/customers/1234 HTTP 1.1
Host: www.example.com
User-Agent:     XYZ 1.1
Accept:         text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Keep-Alive:     300
Connection:     keep-alive
If-Modified-Since:  Fri, 02 Oct 2009 16:47:31 GMT
If-None-Match:  "6[illegible]474f6852c9dab"
Cache-Control: max-age=60

HTTP/1.1 200 OK
Date:  Sun, 04 Oct 2009 19:36:25 GMT
Server:         Apache/2.2.11 (Debian)
Last-Modified:Fri, 02 Oct 2009 16:47:31 GMT
Etag:   "3524acac-59fb-474f6852c9dab"
Cache-Control:  max-age=300
Accept-Ranges:          bytes
Vary:   Accept-Encoding
Content-Encoding:   gzip
Content-Length:         7160
Keep-Alive:     timeout=15, max=91
Connection:   Keep-Alive
Content-Type:           application/xml

<?xml version='1.0' encoding='utf-8' ?>
...
```

getOrderDetails()

submitApplicationData()

updateQuote()

findMatchingBid()

initiateProcess()

cancelSubscription()

listAuctions()

getUsers()

getOrderDetails()

findMatchingBid()

listAuctions()

getUsers()

GET

initiateProcess()

submitApplicationData()

POST

PUT

updateQuote()

DELETE

cancelSubscription()

```
interface Resource {
      Resource(URI u)
      Response get()
      Response post(Request r)
      Response put(Request r)
      Response delete()
}
```

generic

Any HTTP client
(Firefox, IE, curl, wget)

Any HTTP server

Caches

Proxies

Google, Yahoo!, MSN

```
class CustomerCollection : Resource {
      ...
      Response post(Request r) {
            id = createCustomer(r)
            return new Response(201, r)
      }
      ...
}
```

Anything that knows
your app

specific

Google basel II Search Advanced Search Preferences

Web Results **1** - **10** of about **61**

Basel II – Wikipedia - [Translate this page]
Der Terminus **Basel II** bezeichnet die Gesamtheit der Eigenkapitalvorschriften, die vom **Basler** Ausschuss für Bankenaufsicht in den letzten Jahren ...
de.wikipedia.org/wiki/**Basel_II** - 72k - Cached - Similar pages -

Bundesbank - Bankenaufsicht - **Basel II** - [Translate this page]
Basel II - Die neue Baseler Eigenkapitalvereinbarung ... Während der im Jahr 1998 begonnenen Entwicklung des **Basel II** Regelwerkes standen die Aufsicht und ...
www.bundesbank.de/bankenaufsicht/bankenaufsicht_**basel**.php - 27k - Cached - Similar pages -

Bundesbank - Bankenaufsicht - **Basel II** - Säule **2**: Aufsichtliches ... - [Translate this page]
Die drei Saeulen von **Basel 2** Im Rahmen dieser so genannten zweiten Säule, die als integraler Bestandteil des neuen Kapitalakkords gleichberechtigt neben den ...
www.bundesbank.de/bankenaufsicht/bankenaufsicht_**basel**_saeule2.php - 17k - Cached - Similar pages -
More results from www.bundesbank.de »

Basel-II.info - Steuerkanzlei Christian Reichling, Köln - [Translate this page]
Dipl.-Kfm.(FH) Christian Reichling, Steuerberater aus Köln (Cologne) berät Sie gerne in allen Fragen der Bilanzierung und Besteuerung von Gesellschaften ...
www.**basel-ii**.info/ - 12k - Cached - Similar pages -

Definition **Basel 2, Basel II** - [Translate this page]
Ausführliche Informationen zum Thema **Basel II** – neue Bedingungen für Firmenkredite, Rating und Tipps.
www.foerderland.de/353.0.html - 48k - Cached - Similar pages -

Deutscher Industrie- und Handelskammertag / Homepage - [Translate this page]
Der Deutsche Industrie- und Handelskammertag (DIHK) ist die Spitzenorganisation der 80 Industrie- und Handelskammern (IHKs) in Deutschland.
www.dihk.de/inhalt/informationen/news/schwerpunkte/rating/**basel**.html - 6k - Cached - Similar pages -

Unlock information as resources

VersicherungX Hans Müller Search Advanced Search Preferences

Web Results **1** - **10** of about

Versicherungsnehmer: Hans Müller - [Translate this page]
Hans Müller, 40880 Ratingen, Kunde seit dem 1.3.2001, letzte Aktualisierung 1.1.2009 ...
crm.example.com/customers/4711 - 72k - Cached - Similar pages -

Kfz-Haftpflicht Hans Müller - [Translate this page]
Kfz.-Haftpflicht für ME-HM 123, abgeschlossen 1.1.2005, Halter: **Hans Müller**, Audi A4 ...
bestand.example.com/lv/26621 - 72k - Cached - Similar pages -

Risiko LV Anne Müller - [Translate this page]
Risiko LV Anne Müller, 40880 Ratingen, abgeschlossen 1.1.2004, Begünstiger: **Hans Müller**, Versicherungssumme: €200.000 ...
bestand.example.com/lv/26621 - 72k - Cached - Similar pages -

The REST Uniform Interface

identification of resources

resource manipulation through representations

self-descriptive messages

hypermedia as the engine of application state

```
<order self='http://example.com/orders/3321'>
  <amount>23</amount>
  <product ref='http://example.com/products/4554' />
  <customer ref='http://example.com/customers/1234' />
  <link rel='edit’
        ref='http://example.com/order-edit/ACDB' />
</order>
```

Hypermedia

Server provides …

Client follows …

Links

Link relations

“Recipes”

Resources and IDs

Rails 2

Simple:

```
map.resources :orders
```

Nested:

```
map.resources :customers do |customers|
  customers.resources :orders
end
```

Prefixed:

```
map.resources :comments, :path_prefix => '/articles/:article_id'
```

Singleton:

```
map.resource :profile
```

Associations:

```
map.resources :notes, :has_one => :author,
              :has_many => [:comments, :attachments]
```

Rails 3

```
SampleApp::Application.routes.draw do |map|
  resources :blog do
    resource :author
    resources :posts do
      resources :comments, :mentions
    end
  end
end
```

```
                    GET    /blog/:blog_id/author(.:format)                                {:controller=>"authors", :action=>"show"}
                    POST   /blog/:blog_id/author(.:format)                                {:controller=>"authors", :action=>"create"}
                    PUT    /blog/:blog_id/author(.:format)                                {:controller=>"authors", :action=>"update"}
          blog_author DELETE /blog/:blog_id/author(.:format)                                {:controller=>"authors", :action=>"destroy"}
       new_blog_author GET    /blog/:blog_id/author/new(.:format)                            {:controller=>"authors", :action=>"new"}
      edit_blog_author GET    /blog/:blog_id/author/edit(.:format)                           {:controller=>"authors", :action=>"edit"}
                    GET    /blog/:blog_id/posts/:post_id/comments(.:format)               {:controller=>"comments", :action=>"index"}
 blog_post_comments POST   /blog/:blog_id/posts/:post_id/comments(.:format)               {:controller=>"comments", :action=>"create"}
_blog_post_comment GET    /blog/:blog_id/posts/:post_id/comments/new(.:format)           {:controller=>"comments", :action=>"new"}
                    GET    /blog/:blog_id/posts/:post_id/comments/:id(.:format)           {:controller=>"comments", :action=>"show"}
                    PUT    /blog/:blog_id/posts/:post_id/comments/:id(.:format)           {:controller=>"comments", :action=>"update"}
  blog_post_comment DELETE /blog/:blog_id/posts/:post_id/comments/:id(.:format)           {:controller=>"comments", :action=>"destroy"}
t_blog_post_comment GET    /blog/:blog_id/posts/:post_id/comments/:id/edit(.:format)      {:controller=>"comments", :action=>"edit"}
                    GET    /blog/:blog_id/posts/:post_id/mentions(.:format)               {:controller=>"mentions", :action=>"index"}

  # skipping ...

                    GET    /blog/:blog_id/posts(.:format)                                 {:controller=>"posts", :action=>"index"
        blog_posts POST   /blog/:blog_id/posts(.:format)                                 {:controller=>"posts", :action=>"create"}
     new_blog_post GET    /blog/:blog_id/posts/new(.:format)                             {:controller=>"posts", :action=>"new"}
                    GET    /blog/:blog_id/posts/:id(.:format)                             {:controller=>"posts", :action=>"show"}
                    PUT    /blog/:blog_id/posts/:id(.:format)                             {:controller=>"posts", :action=>"update"}
          blog_post DELETE /blog/:blog_id/posts/:id(.:format)                             {:controller=>"posts", :action=>"destroy"}
    edit_blog_post GET    /blog/:blog_id/posts/:id/edit(.:format)                        {:controller=>"posts", :action=>"edit"}
                    GET    /blog(.:format)                                                {:controller=>"blogs", :action=>"index"}
             blogs POST   /blog(.:format)                                                {:controller=>"blogs", :action=>"create"}
          new_blog GET    /blog/new(.:format)                                            {:controller=>"blogs", :action=>"new"}
                    GET    /blog/:id(.:format)                                            {:controller=>"blogs", :action=>"show"}
                    PUT    /blog/:id(.:format)                                            {:controller=>"blogs", :action=>"update"}
              blog DELETE /blog/:id(.:format)                                            {:controller=>"blogs", :action=>"destroy"}
         edit_blog GET    /blog/:id/edit(.:format)                                       {:controller=>"blogs", :action=>"edit"}
```

```
map.connect ':controller/:action/:id'
```

Resources and IDs

Explicit control over every URI aspect

Low-level connection to Rack apps

High-level resource convenience functions

Single route definition

Functions to build URIs

Formats

Rails 2

```ruby
class CompaniesController < ApplicationController
    def index
    @companies = Company.all
    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @companies }
      format.json { render :json => @companies }
    end
  end

  def show
    @company = Company.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @company }
      format.json { render :json => @company }
    end
  end
```

Rails 3

```
class CompaniesController < ApplicationController::Base
  respond_to :html, :xml, :json

  def index
    respond_with(@companies = Company.all)
  end

  def new
    respond_with(@company = Company.new)
  end

  def create
    respond_with(@company = Company.create(params[:company]))
  end

  def edit
    respond_with(@company = Company.find(params[:id]))
  end

  def update
    @company = Company.find(params[:id])
    @company.update_attributes(params[:company])
    respond_with(@company)
  end
end
```

Formats

Decoupling of resource and representation

Support for content negotiation

Custom responders for own formats

Override where needed

Self-descriptive messages

Self-descriptive messages

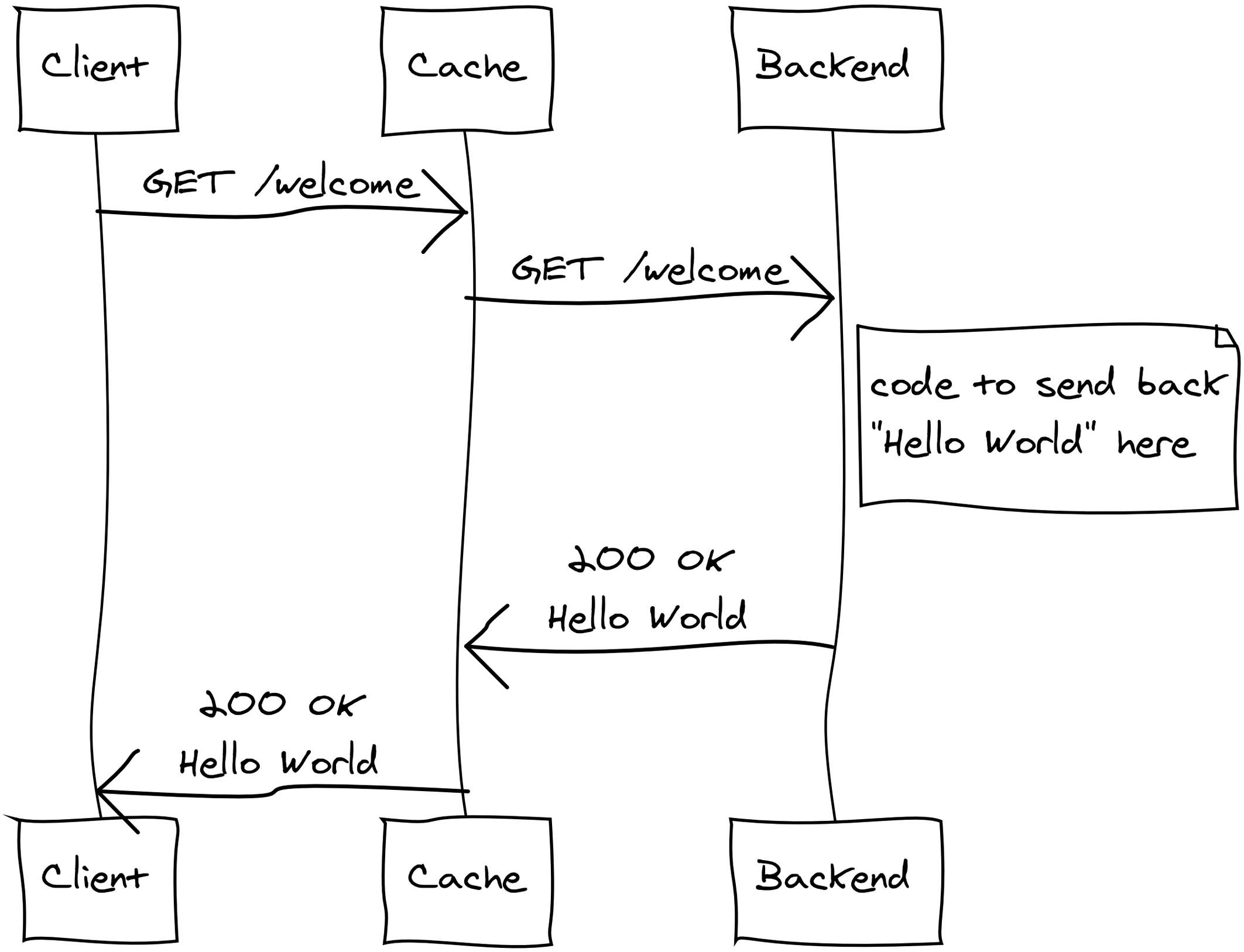
Standard methods

Caching

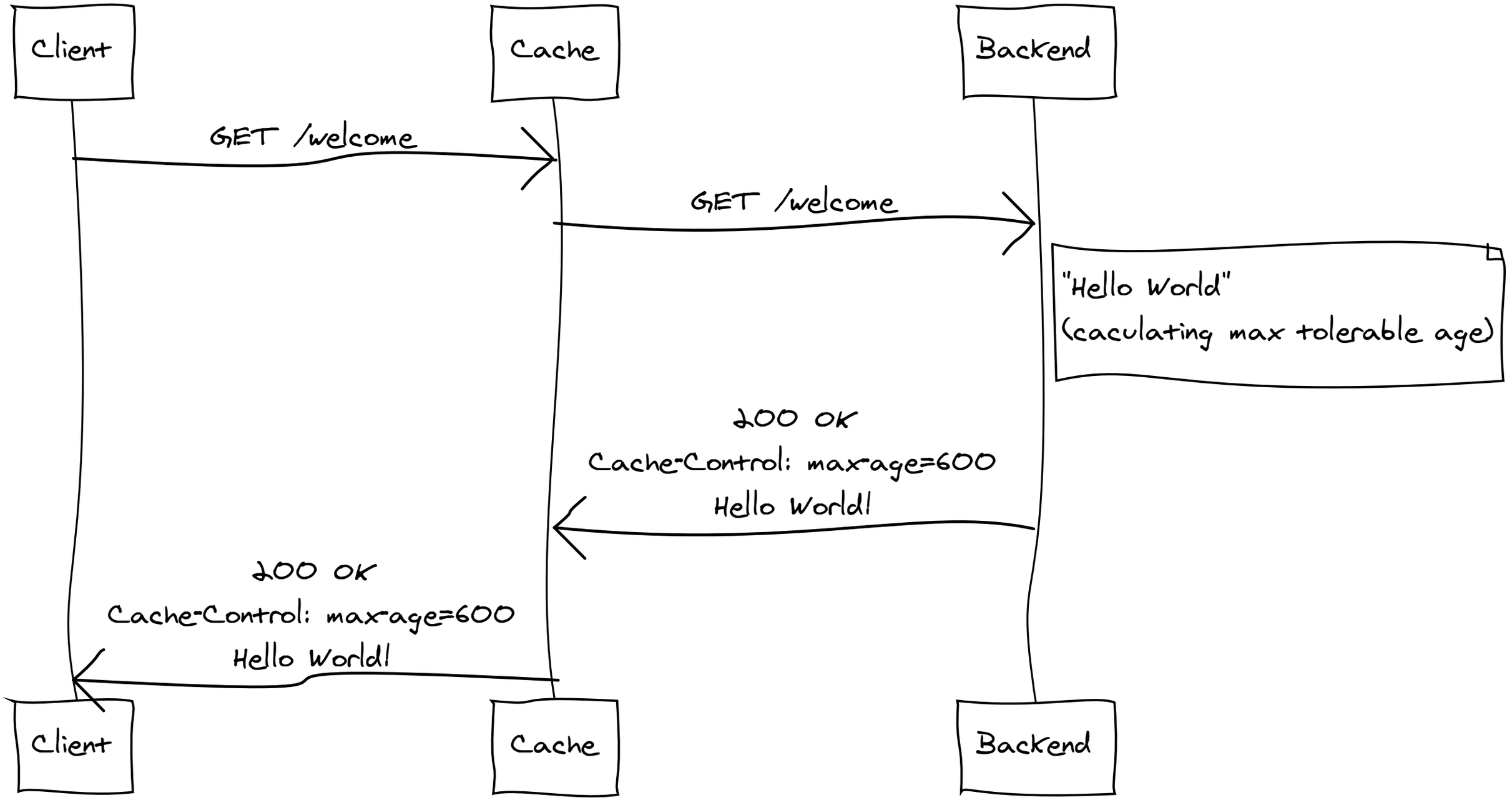
Conditional requests

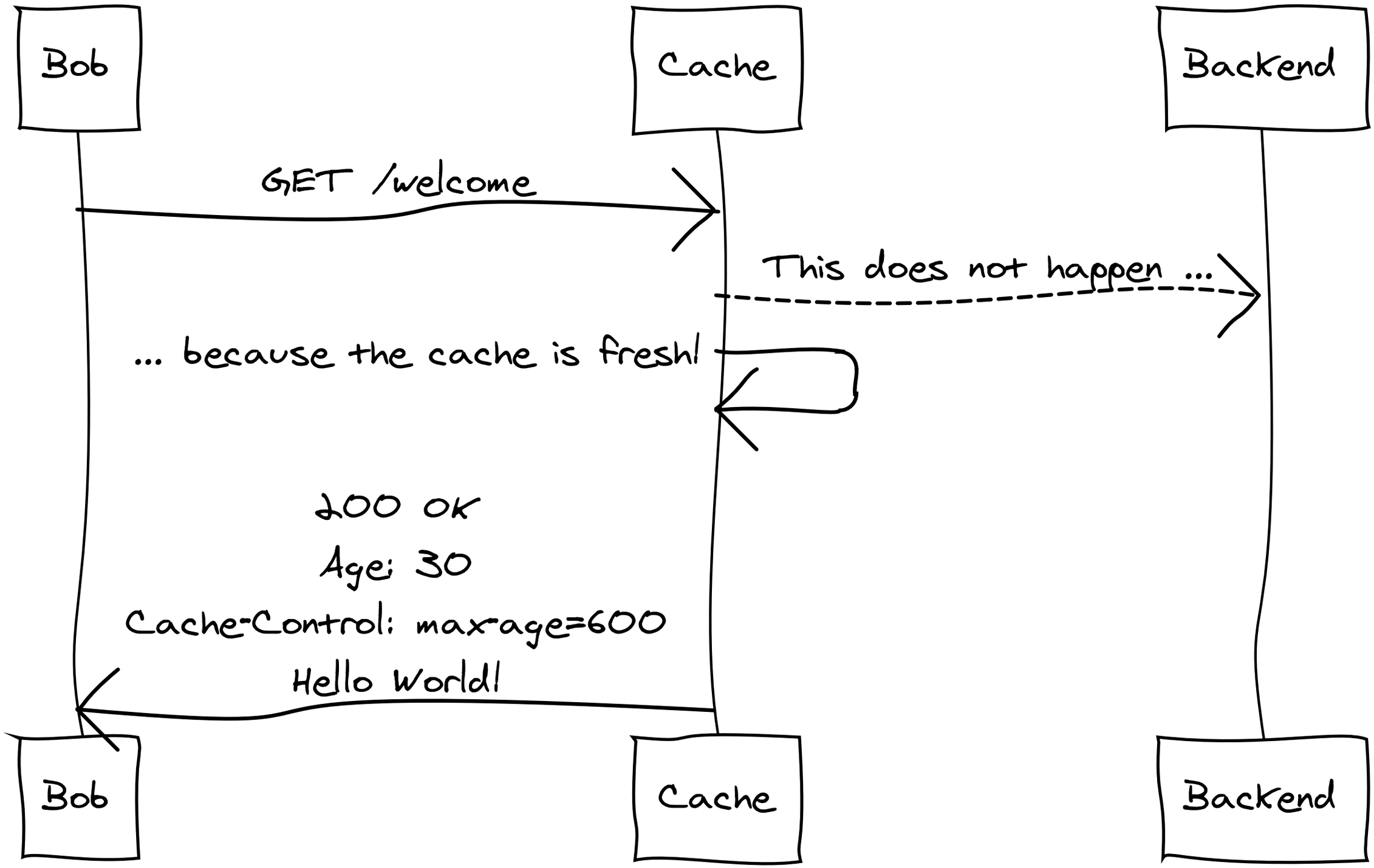
Caching Models

Note: Thanks to Ryan Tomayko for letting me ~~steal~~ base some work on his diagrams from http://tomayko.com/writings/things-caches-do

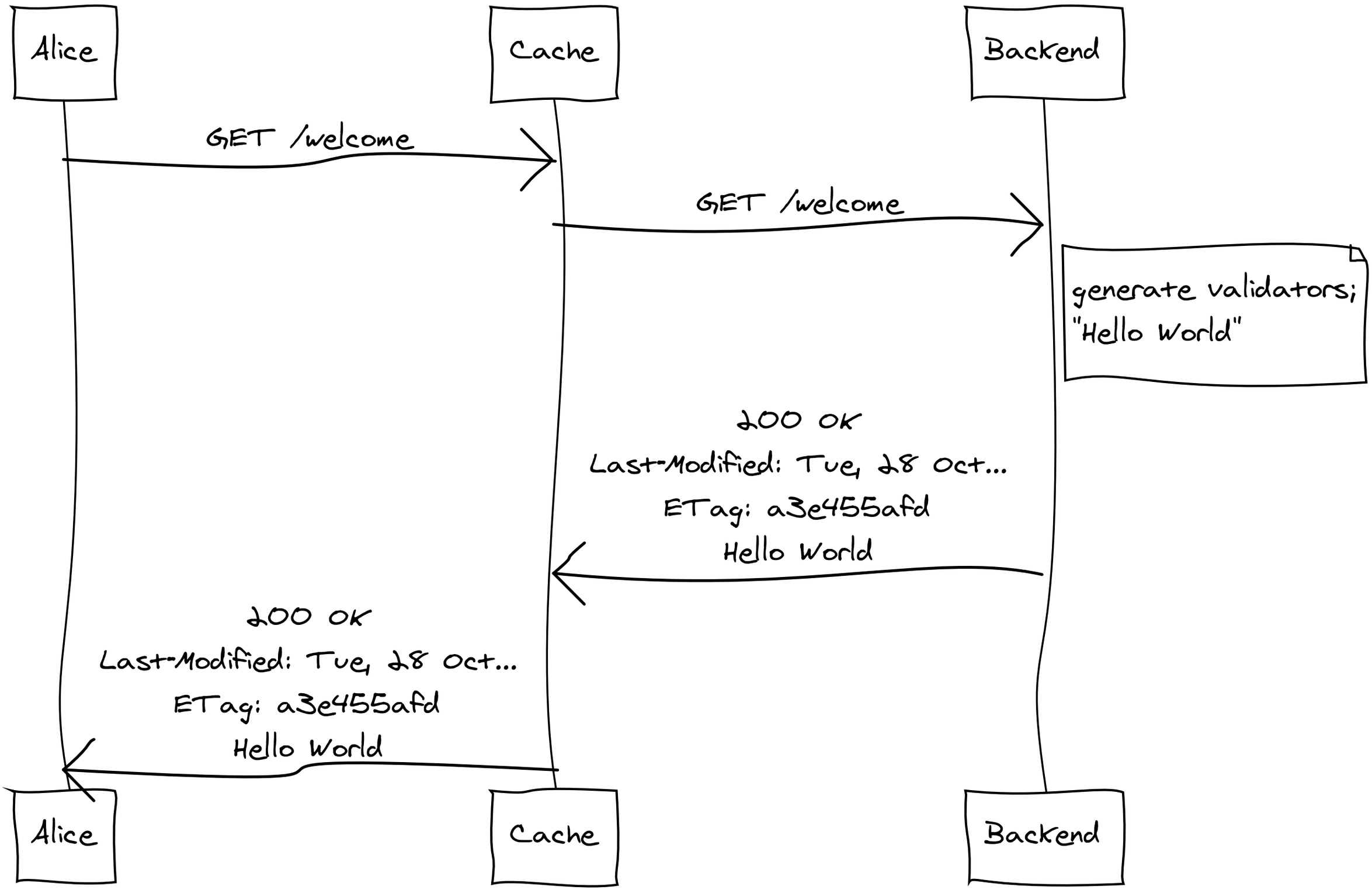

Client
Cache
Backend
GET /welcome
GET /welcome
code to send back
"Hello World" here
200 OK
Hello World
200 OK
Hello World
Client
Cache
Backend

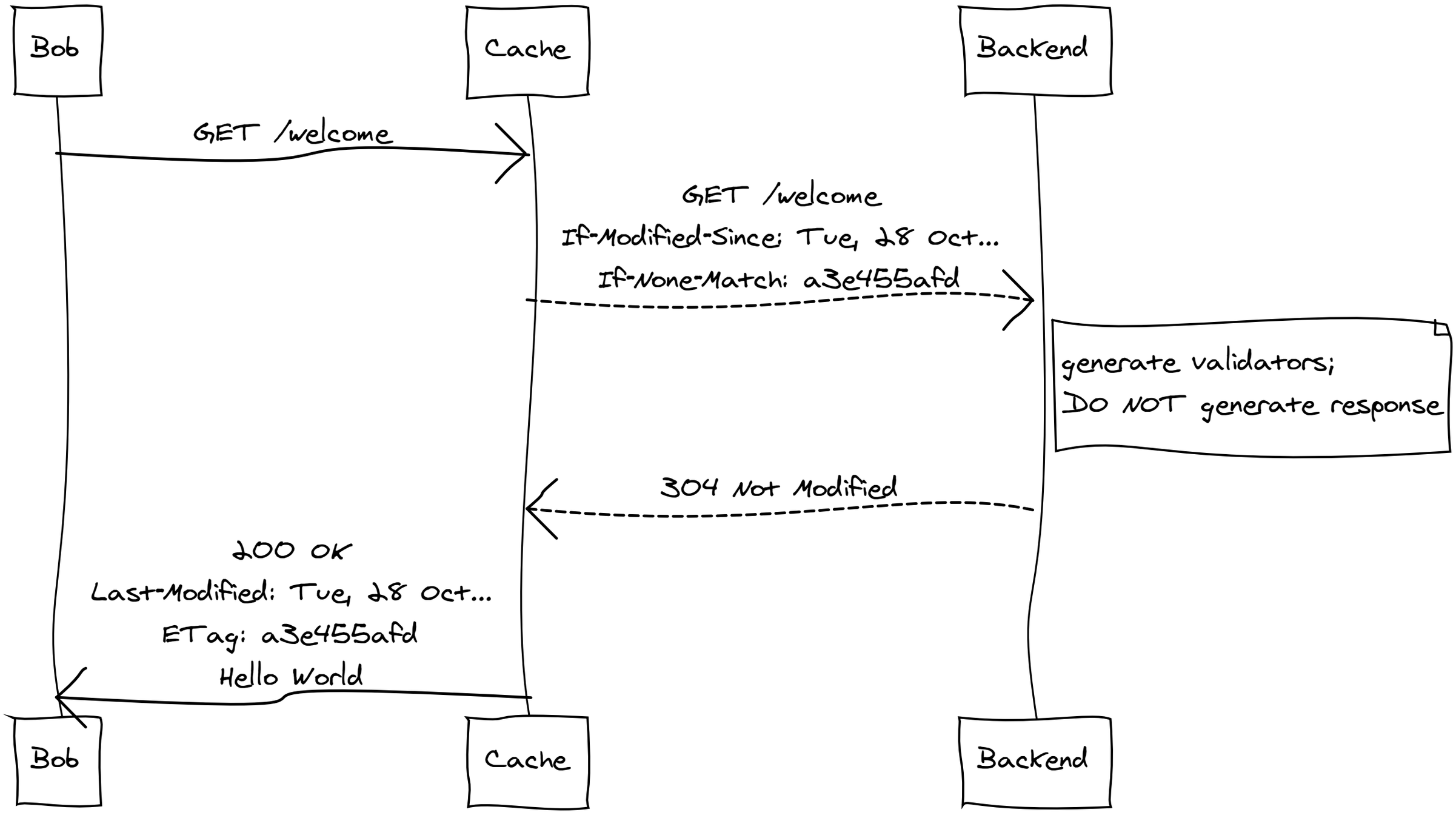
Expiration


Client
Cache
Backend
GET /welcome
GET /welcome
"Hello World"
(caculating max tolerable age)
200 OK
Cache-Control: max-age=600
Hello World!
200 OK
Cache-Control: max-age=600
Hello World!
Client
Cache
Backend

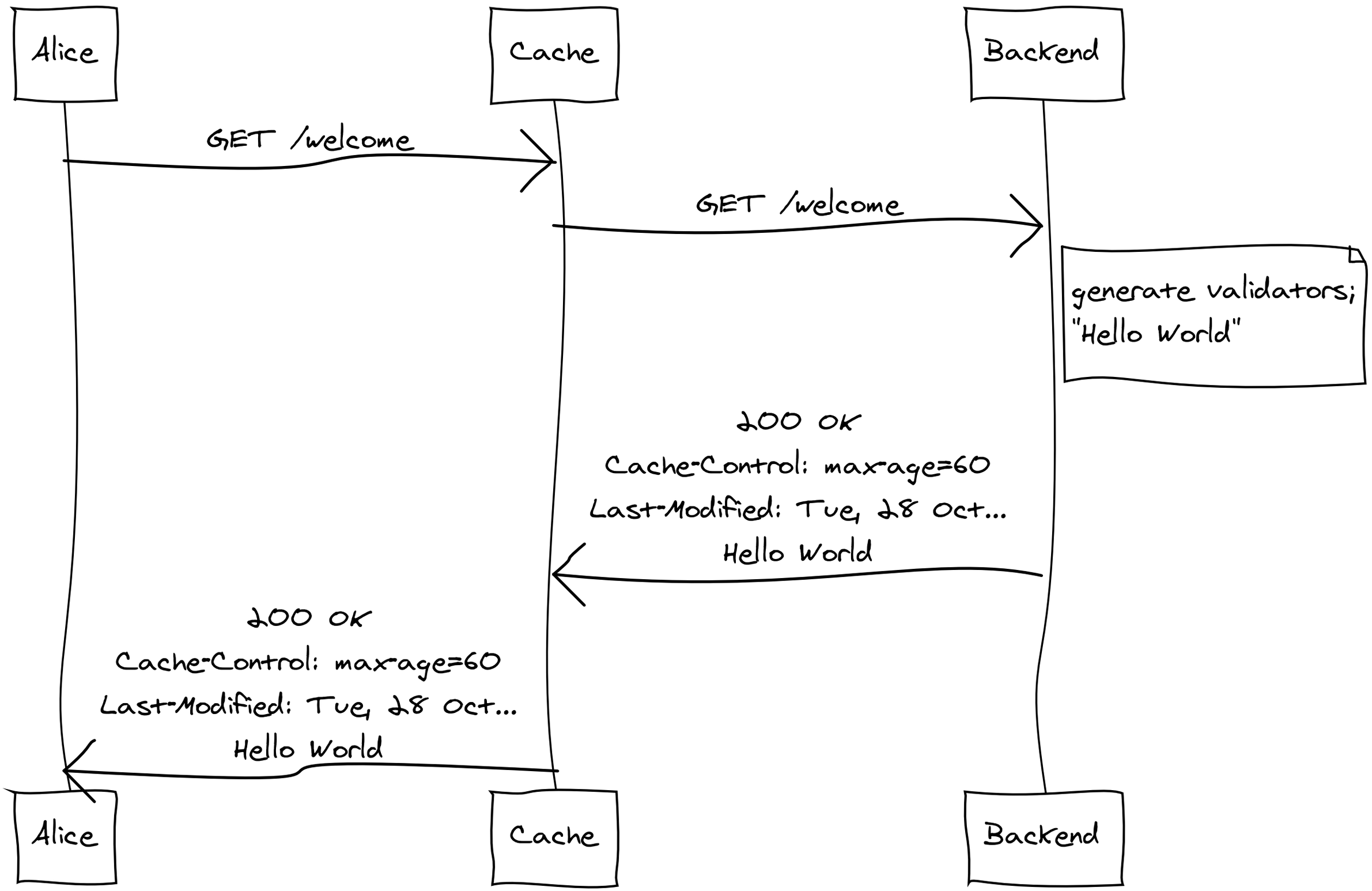

Bob
Cache
Backend
GET /welcome
This does not happen ...
... because the cache is fresh!
200 OK
Age: 30
Cache-Control: maxage=600
Hello World!
Bob
Cache
Backend

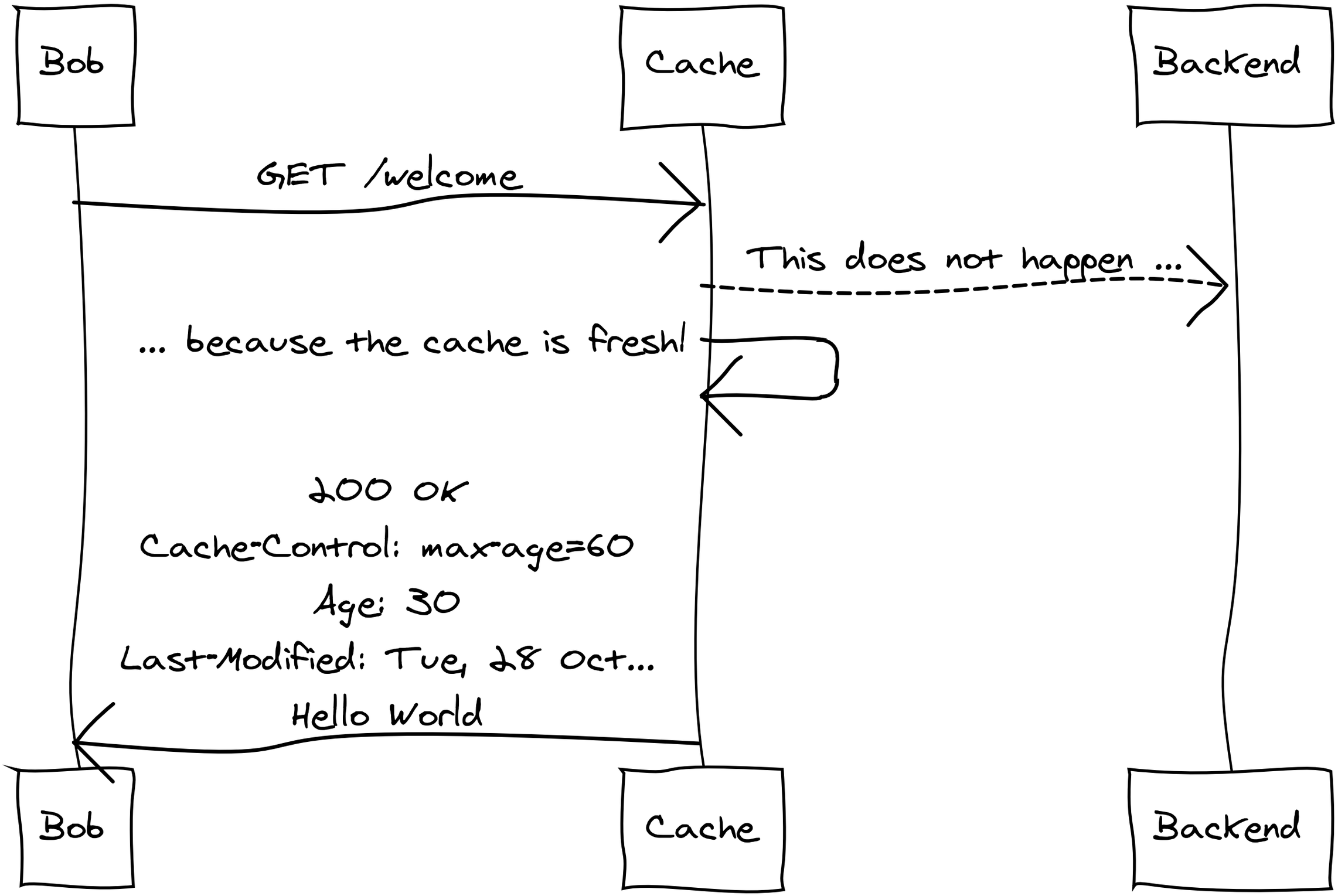
Validation

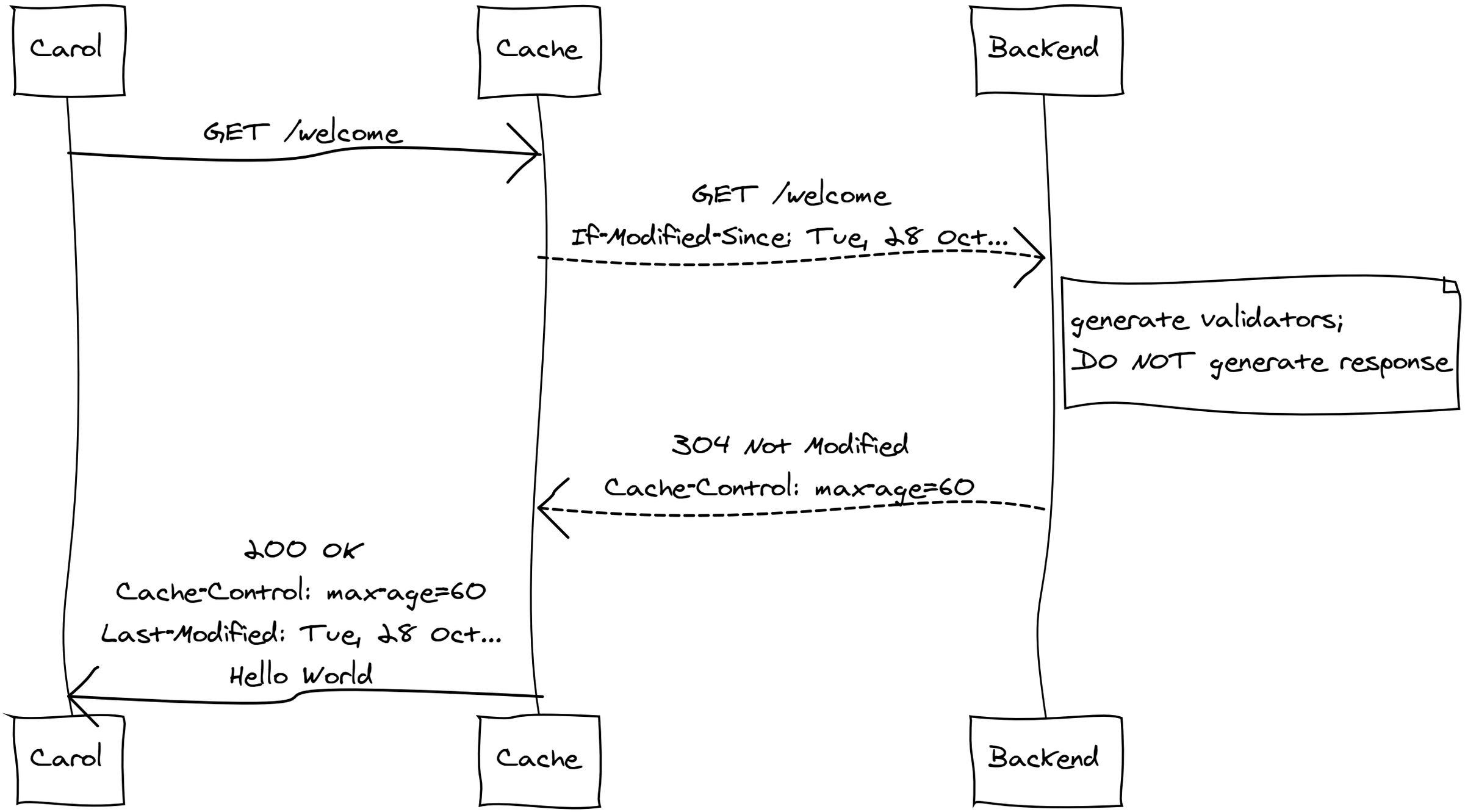

Alice
Cache
Backend
GET /welcome
GET /welcome
generate validators;
"Hello World"
200 OK
Last-Modified: Tue, 28 Oct...
ETag: a3e455afd
Hello World
200 OK
Last-Modified: Tue, 28 Oct...
ETag: a3e455afd
Hello World
Alice
Cache
Backend


Bob
Cache
Backend
GET /welcome
GET /welcome
If-Modified-Since: Tue, 28 Oct...
If-None-Match: a3e455afd
generate validators;
DO NOT generate response
304 Not Modified
200 OK
Last-Modified: Tue, 28 Oct...
ETag: a3e455afd
Hello World
Bob
Cache
Backend

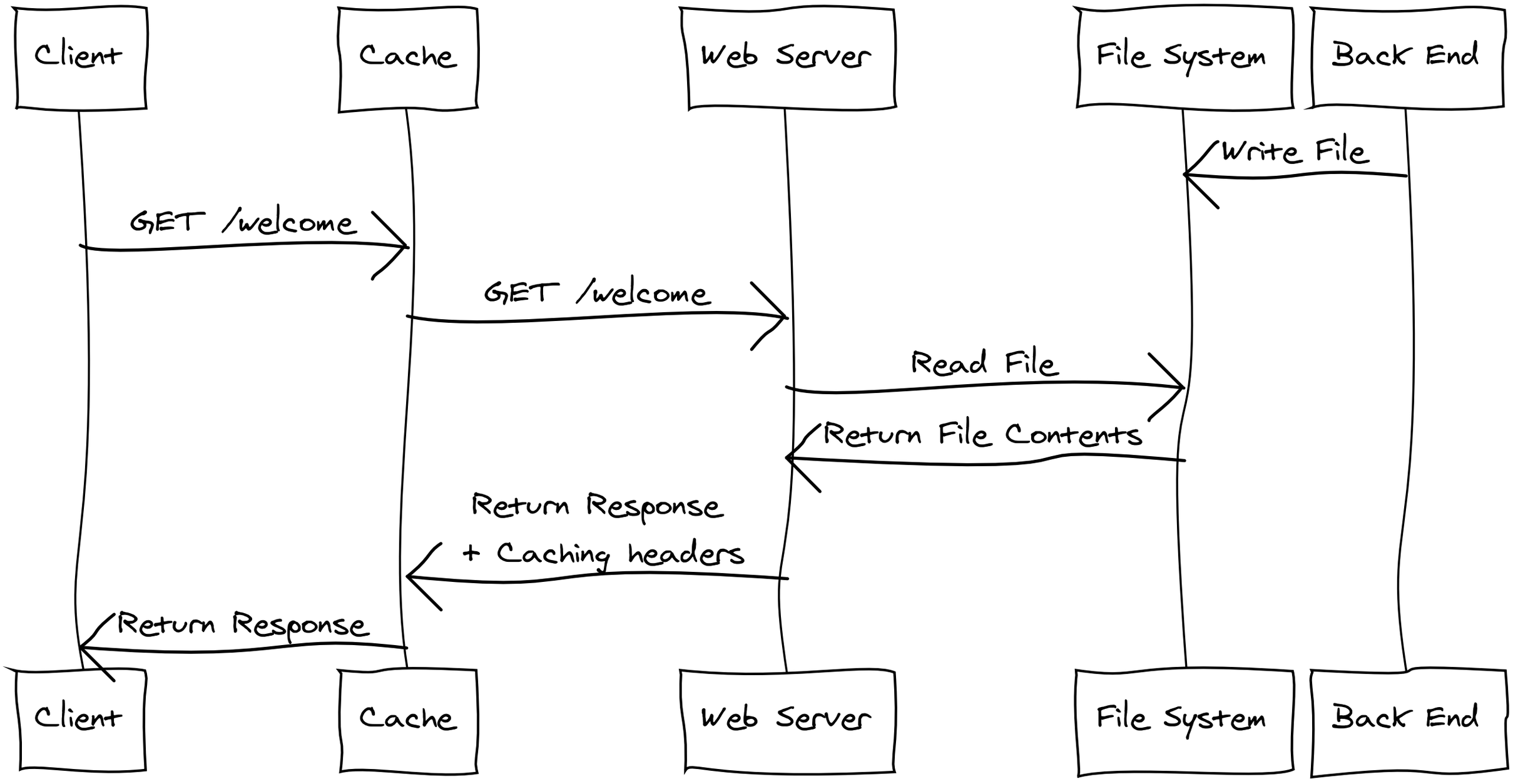
Combination


Alice
Cache
Backend
GET /welcome
GET /welcome
generate validators;
"Hello World"
200 OK
Cache-Control: max-age=60
Last-Modified: Tue, 28 Oct...
Hello World
200 OK
Cache-Control: max-age=60
Last-Modified: Tue, 28 Oct...
Hello World
Alice
Cache
Backend

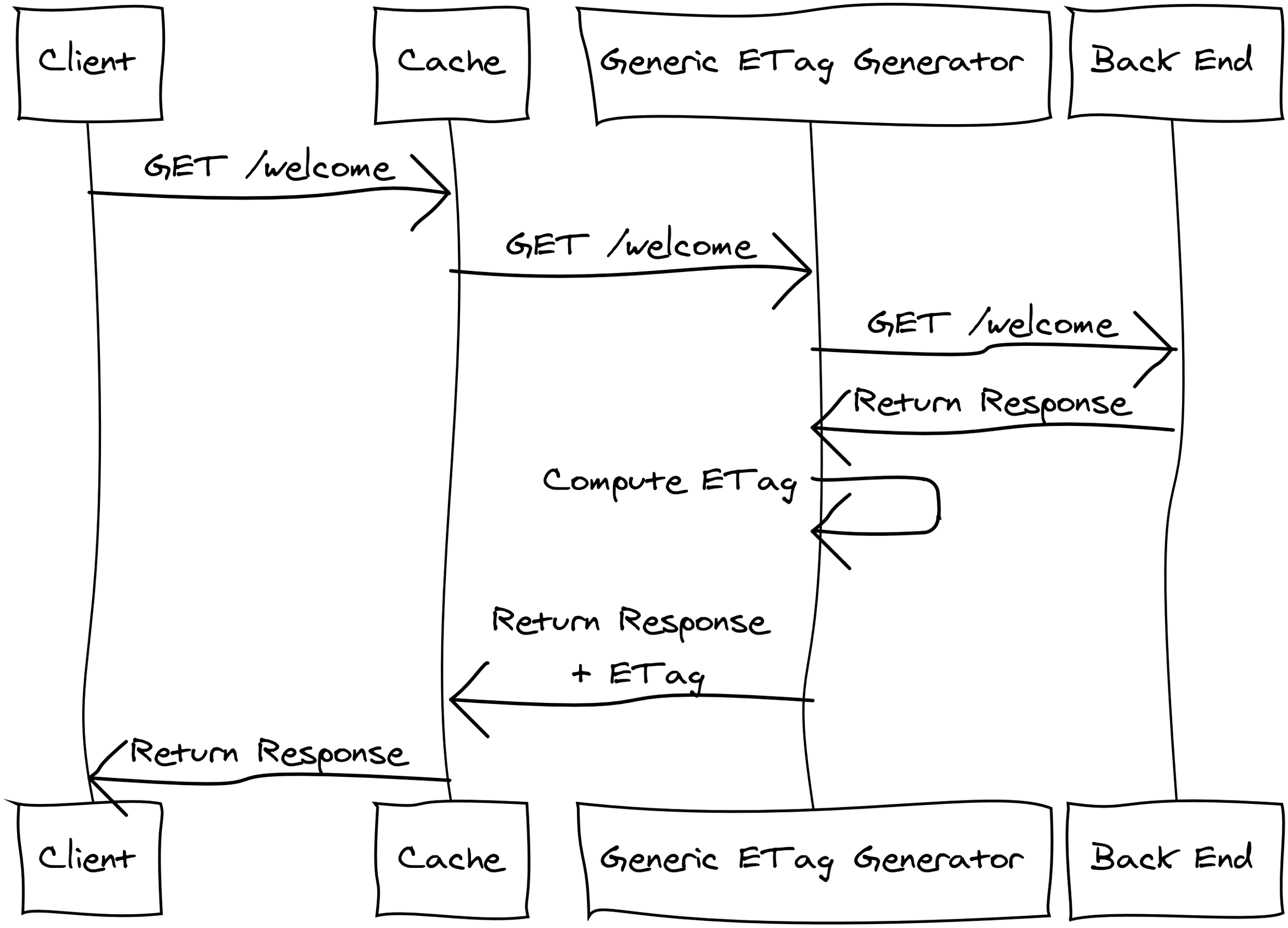

Bob
Cache
Backend
GET /welcome
This does not happen ...
... because the cache is fresh!
200 OK
Cache-Control: max-age=60
Age: 30
Last-Modified: Tue, 28 Oct...
Hello World
Bob
Cache
Backend

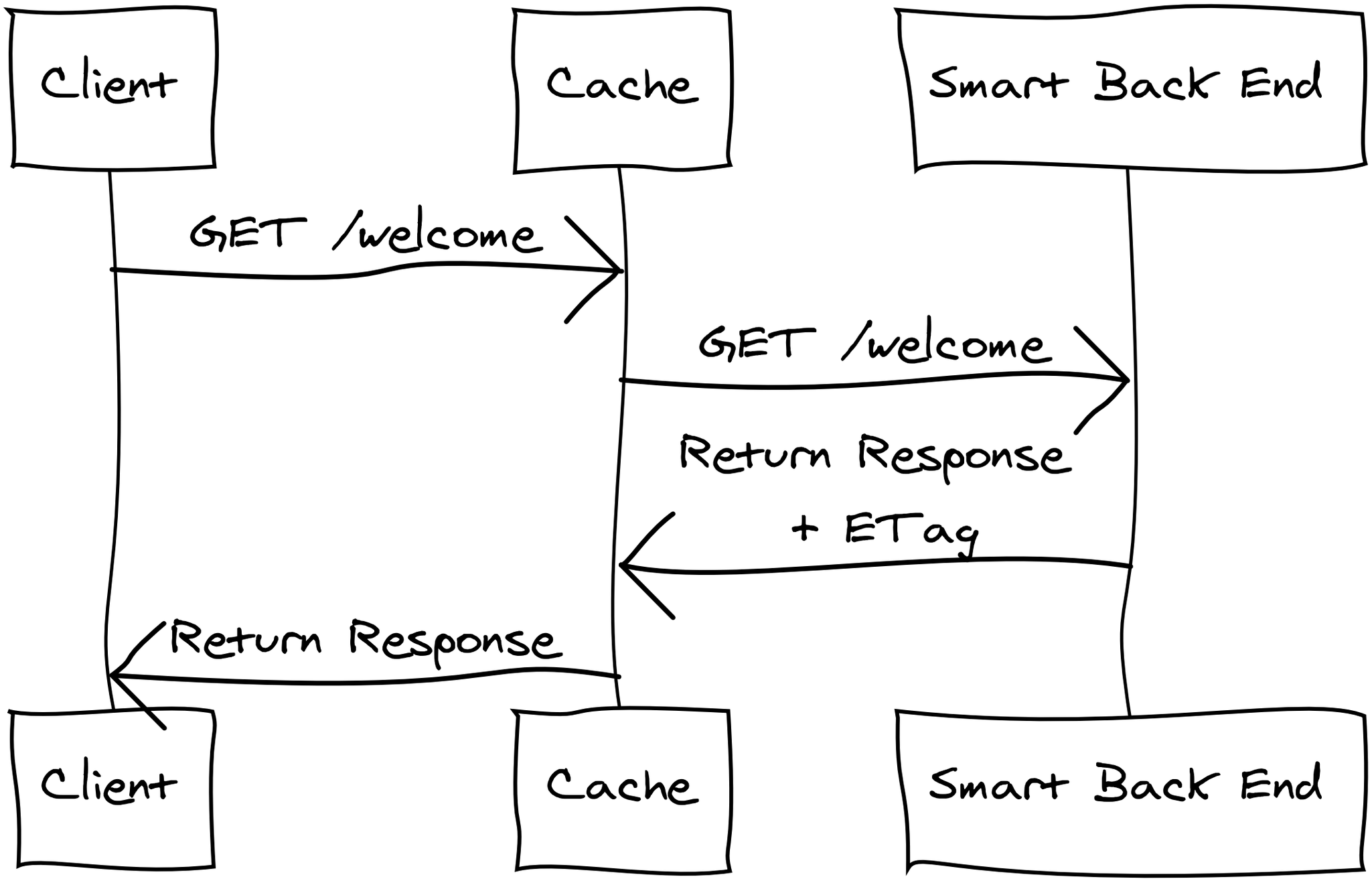

Carol
Cache
Backend
GET /welcome
GET /welcome
If-Modified-Since: Tue, 28 Oct...
generate validators;
DO NOT generate response
304 Not Modified
Cache-Control: max-age=60
200 OK
Cache-Control: max-age=60
Last-Modified: Tue, 28 Oct...
Hello World
Carol
Cache
Backend

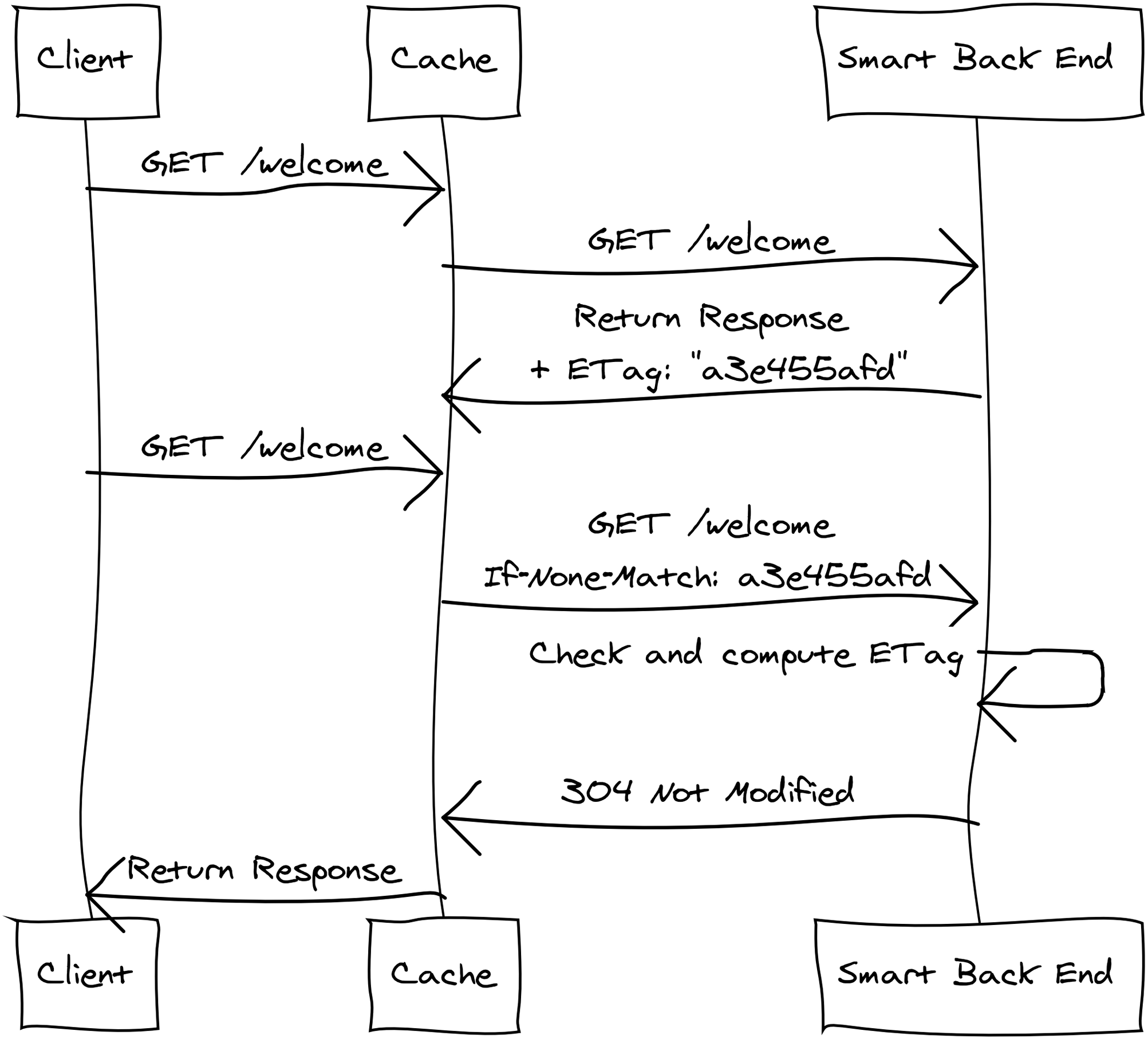
Trivial Implementation


Client
Cache
Web Server
File System
Back End
Write File
GET /welcome
GET /welcome
Read File
Return File Contents
Return Response
+ Caching headers
Return Response
Client
Cache
Web Server
File System
Back End

ETag Depth


Client
Cache
Generic ETag Generator
Back End
GET /welcome
GET /welcome
GET /welcome
Return Response
Compute ETag
Return Response
+ ETag
Return Response
Client
Cache
Generic ETag Generator
Back End


Client
Cache
Smart Back End
GET /welcome
GET /welcome
Return Response
+ ETag
Return Response
Client
Cache
Smart Back End


Client
Cache
Smart Back End
GET /welcome
GET /welcome
Return Response
+ ETag: "a3e455afd"
GET /welcome
GET /welcome
If-None-Match: a3e455afd
Check and compute ETag
304 Not Modified
Return Response
Client
Cache
Smart Back End

Rails, ETags, Cache headers

Built-in "shallow" ETag generation

Explicit controller API methods

expires_in, expires_now

fresh_when

stale?

A little less shallow …

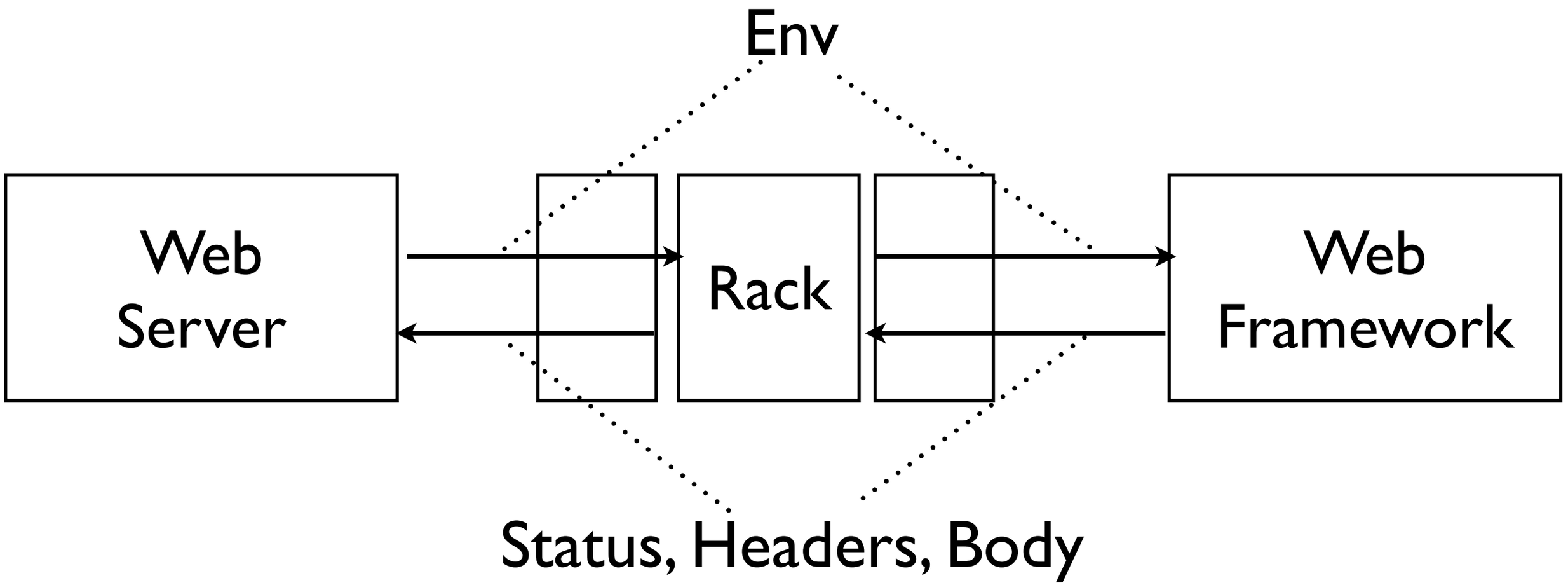
```ruby
def show
  @company = Company.find(params[:id])
  expires_in 30.seconds
  fresh_when(:etag => @company, :last_modified => @company.created_at.utc,
             :public => true)
  respond_to do |format|
    format.html # show.html.erb
    format.xml  { render :xml => @company }
    format.json { render :json => @company }
  end
end
```

Deep ETags

```
def index
  expires_in 60.seconds
  if stale?(:last_modified => Company.maximum('updated_at').utc,
            :etag => calculate_something_quickly, :public => true)
    @companies = Company.all
    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @companies }
      format.json { render :json => @companies }
    end
  end
end
```

Rack

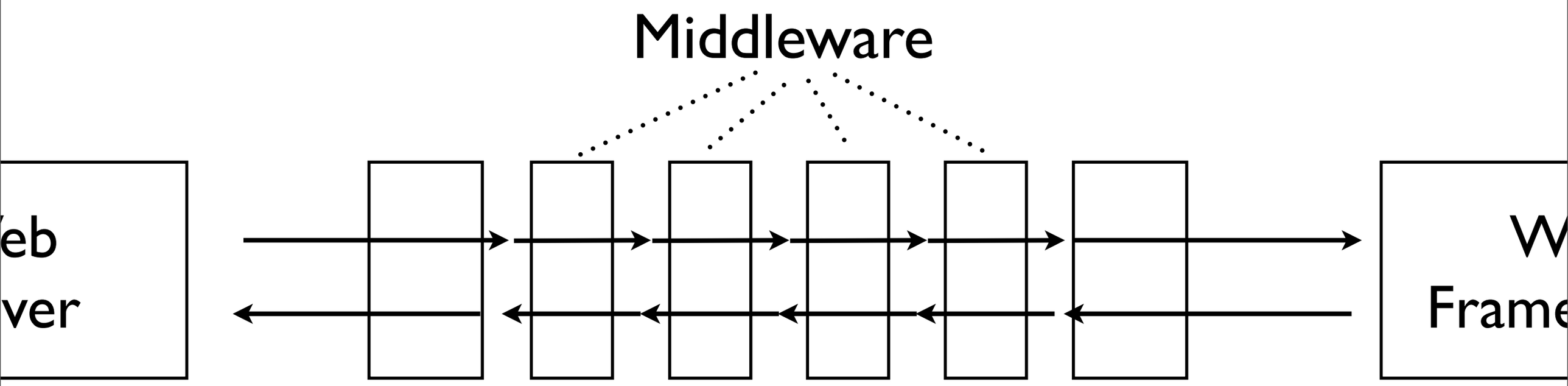
Decoupling through Rack



Minimal Rack Handler

```
class HelloApp
  def call(env)
    [200,
     {"Content-Type" => "text/plain"},
     "Hello, world.\n\n"]
  end
end

map '/' do
run MinimalRackApp.new
end
```



Cascade
Chunked
CommonLogger
ConditionalGet
Config
ContentLength
ContentType
ETag
File
Deflater
Directory
ForwardRequest
Handler
Head
Lint
Lock
Logger
MethodOverride
Mime
NullLogger
Recursive
Reloader
Runtime
Sendfile
Server
ShowExceptions
ShowStatus
Static
URLMap
MockRequest
MockResponse
Auth::Basic
Auth::Digest
Session::Cookie
Session::Pool
Session::Memcache

Rack Benefits

Simplest possible API

Existing middleware

Extensive integration w/ servers and frameworks

Rails support for multiple apps, potentially built with multiple frameworks

Self-descriptive Messages

Low-level access to every HTTP aspect

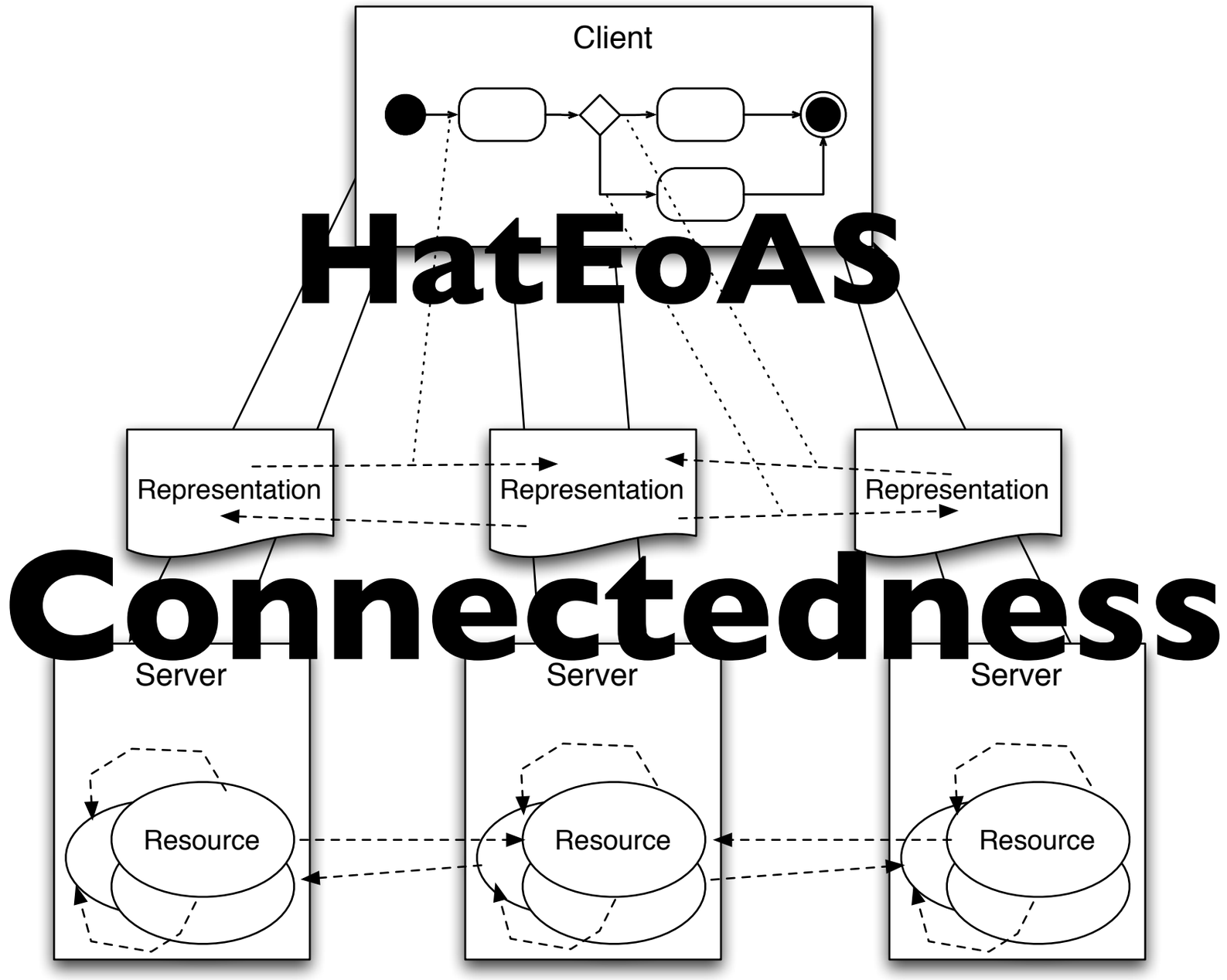
Controller-side abstractions

Built-in generic solutions

Hypermedia

?

Hypermedia levels



Clients

ActiveResource

Non-generic

Tied to Rails models

Non-standard format

Tightly coupled

Net::HTTP

RestClient

HTTParty

Curl
libcurl
Easy
Multi

Patron
Curb
Typhoeus

httpclient

EM-HTTP-Client

Restfully

Restfulie

Restfulie

Client & Server

Hypermedia-focused

Pluggable HTTP providers

Restfulie client example

```
list = Restfulie.at('http://om.example.com/products')
                .as('application/xml').get
```

```
<list>
  <link rel="orders" ref="http://om.example.com/orders" />
  <items>
    <item>
      <descriptions>Cool product ...</descriptions>
      <price>25.0</price>
    </item>
  </items>
</list>
```

```
basket = {:items => [{:id => list.item[0][:id]}]}

basket = list.orders.post!(basket.to_xml(:root => 'basket'))
```

```
<basket>
  <link rel="payment" ref="http://om.example.com/orders" />
  ...
</basket>
```

```
payment = {:card_holder => "Guilherme Silveira",
           :cardnumber => '123000004',
           :amount => basket.price }

receipt = basket.payment.post!(payment.to_xml(:root => 'payment'))
```

Restfulie Client Agent Framework

Goal-oriented instead of procedural

Reduces coupling

Restfulie Server Framework

Based on Rails

Backports respond_with

Extensive link support

Support for Atom

Summary

identification of resources

resource manipulation through representations

self-descriptive messages

hypermedia as the engine of application state

3/4

rack

ActiveResource

Lots of options

Lots to explore

A bold statement:

Rails and other Ruby frameworks continue to define how good REST support can be.

Have fun!

Thank you!
Any questions?

http://www.innoq.com

Stefan Tilkov

http://www.innoq.com/blog/st/

innoQ

Architectural Consulting

SOA	WS-*	REST
MDA	MDSD	MDE
J(2)EE	RoR	.NET

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 2102 77 162-100
info@innoq.com · www.innoq.com

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

Resources

Nothingham, Mark: The State of Proxy Caching
http://www.mnot.net/blog/2007/06/20/proxy_caching

Nottingham, Mark: The State of Browser Caching
http://www.mnot.net/blog/2006/05/11/browser_caching

Nottingham, Mark: Caching Tutorial
http://www.mnot.net/cache_docs/

Squid, http://www.squid-cache.org/

Nottingham, Mark: Cache Channels for Squid
http://www.mnot.net/cache_channels/

The Varnish Project, http://varnish.projects.linpro.no/

Varnish: Notes from the Architect
http://varnish.projects.linpro.no/wiki/ArchitectNotes

Tomoyako, Ryan: Things Caches Do
http://tomayko.com/writings/things-caches-do

Rack in Rails 3
http://en.oreilly.com/railswinter10/public/schedule/detail/13304

W3C, ESI Language Specification 1.0
http://www.w3.org/TR/esi-lang

Apache HTTP 2.2 Caching Guide
http://httpd.apache.org/docs/2.2/caching.html

RestClient
http://github.com/archiloque/rest-client

HTTParty
http://github.com/jnunemaker/httparty

httpclient
http://github.com/nahi/httpclient

Restfully
http://github.com/crohr/restfully

Restfulie
http://github.com/caelum/restfulie

EM-HTTP-Client
http://github.com/igrigorik/em-http-request

Typhoeus
http://github.com/pauldix/typhoeus

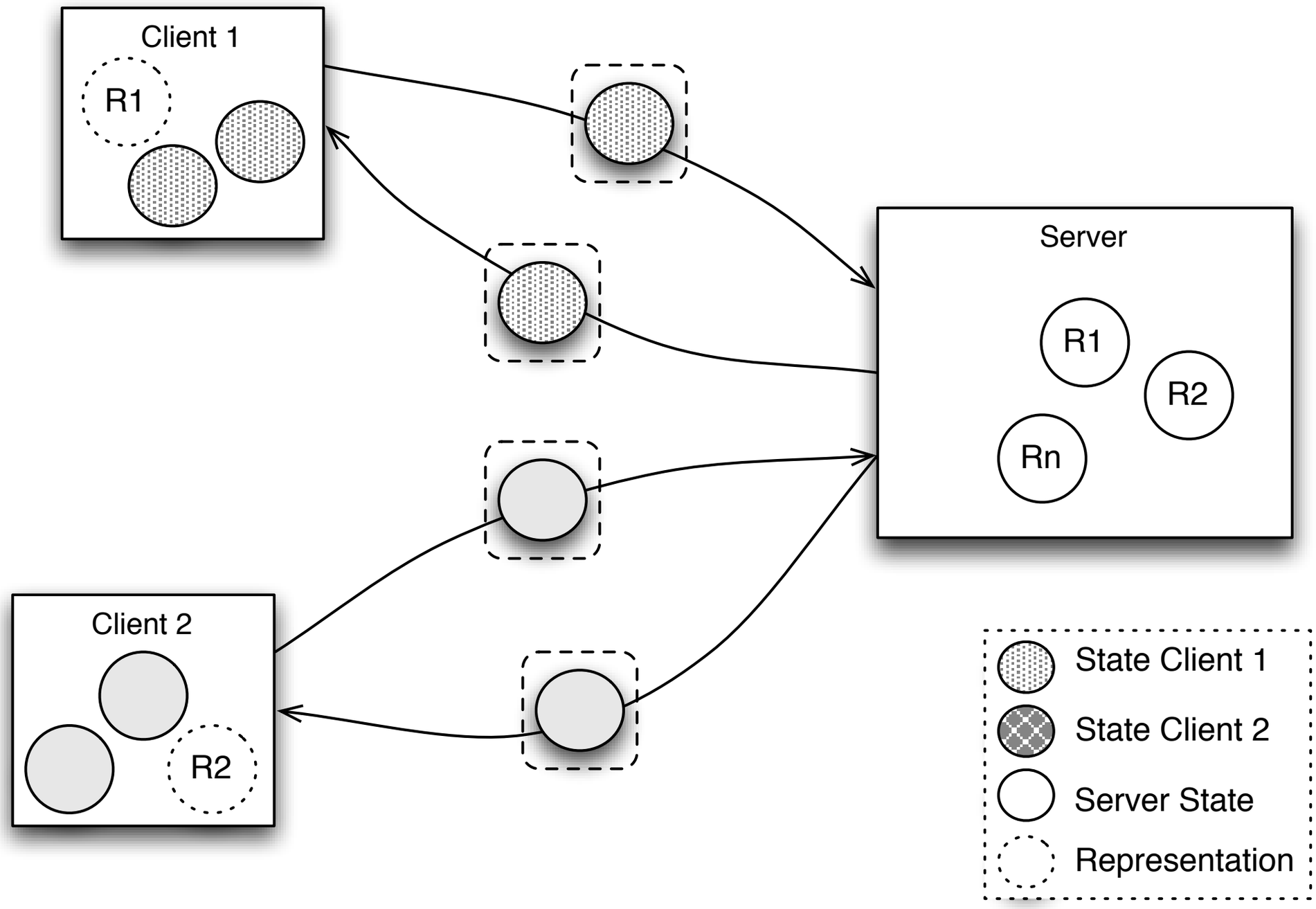
Patron
http://github.com/toland/patron

Curb
http://github.com/taf2/curb

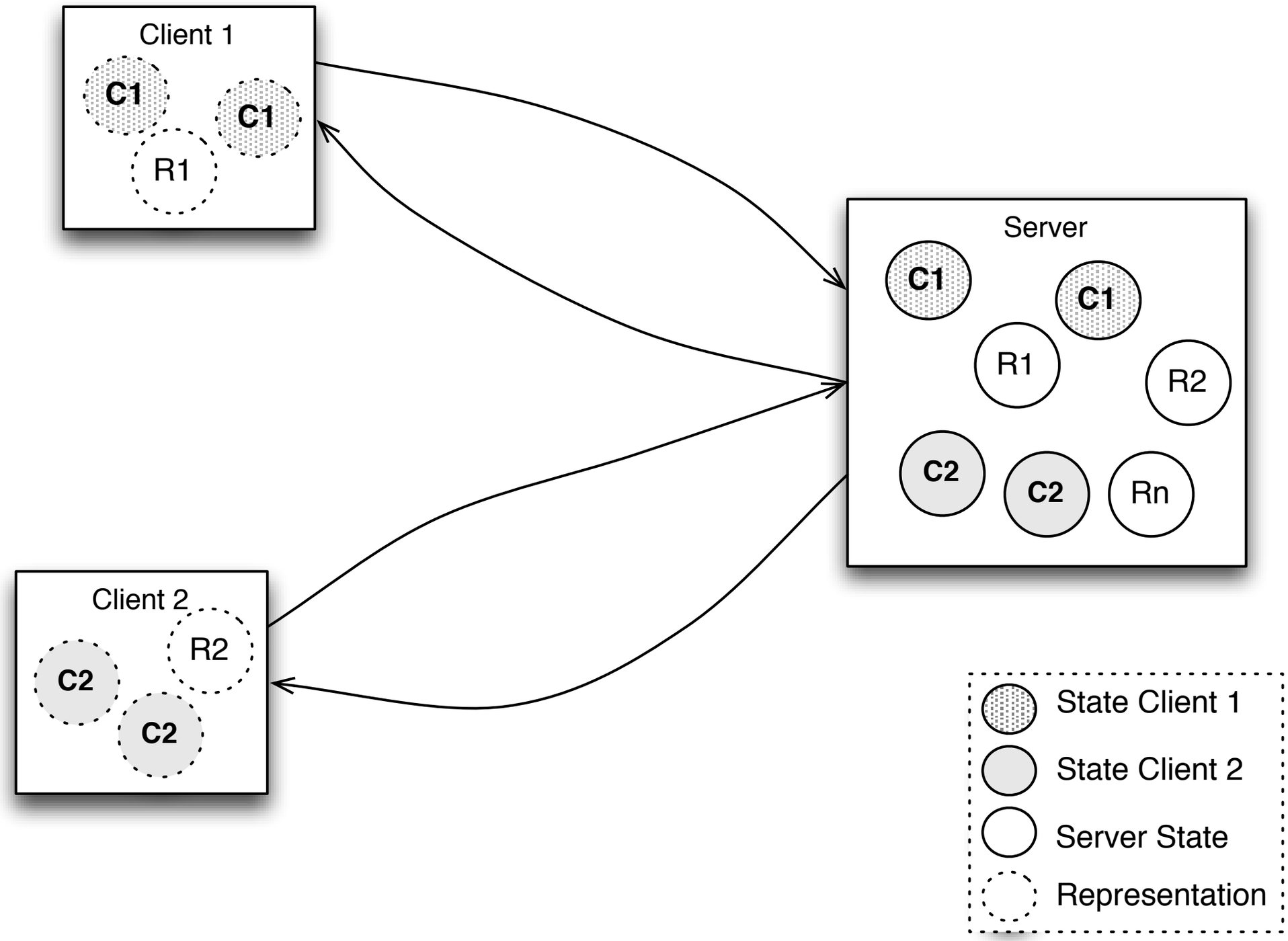
Backup

State

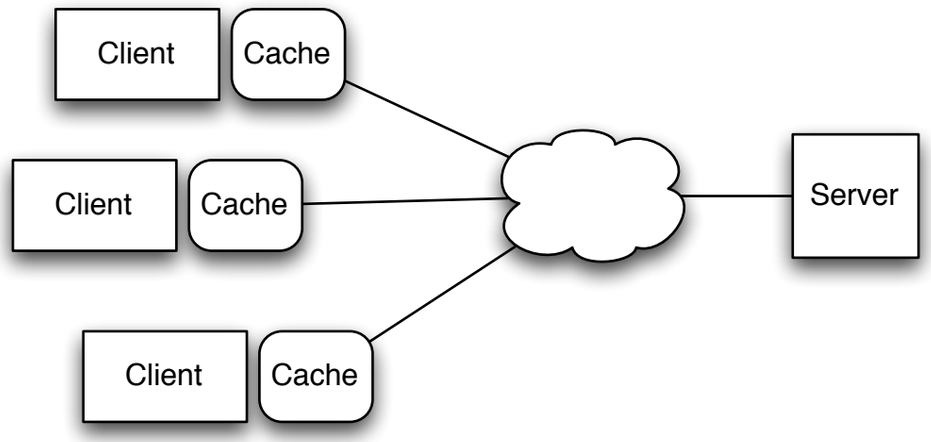
Turning Session State …



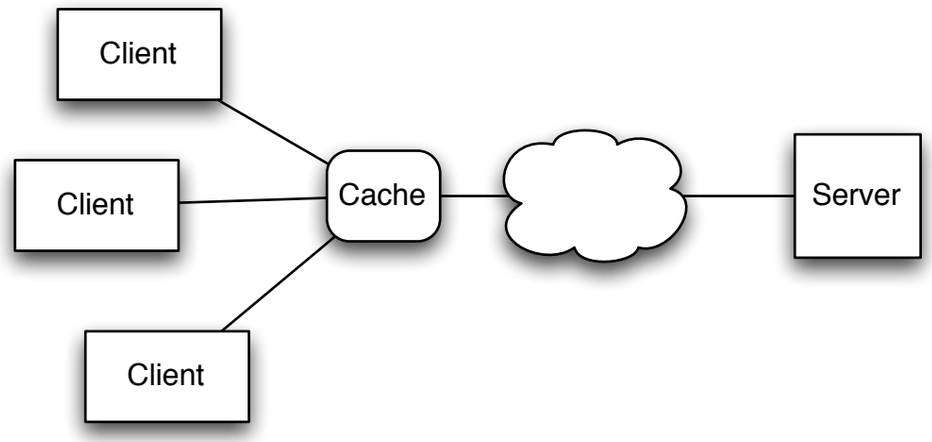
… into Resource or Client State



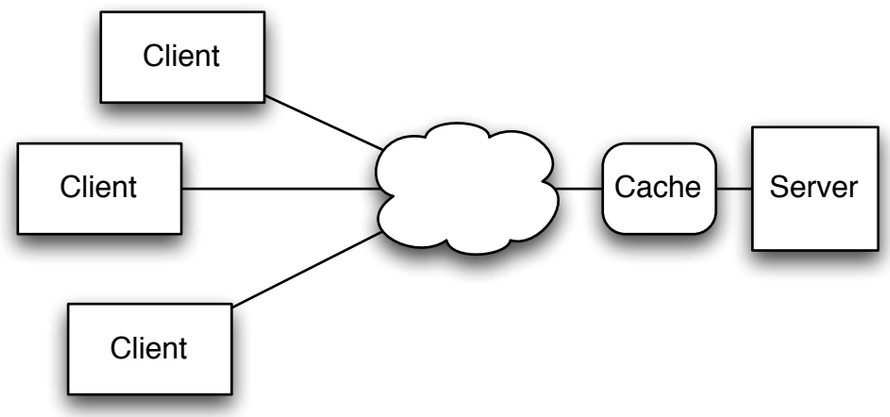
Cache Topologies



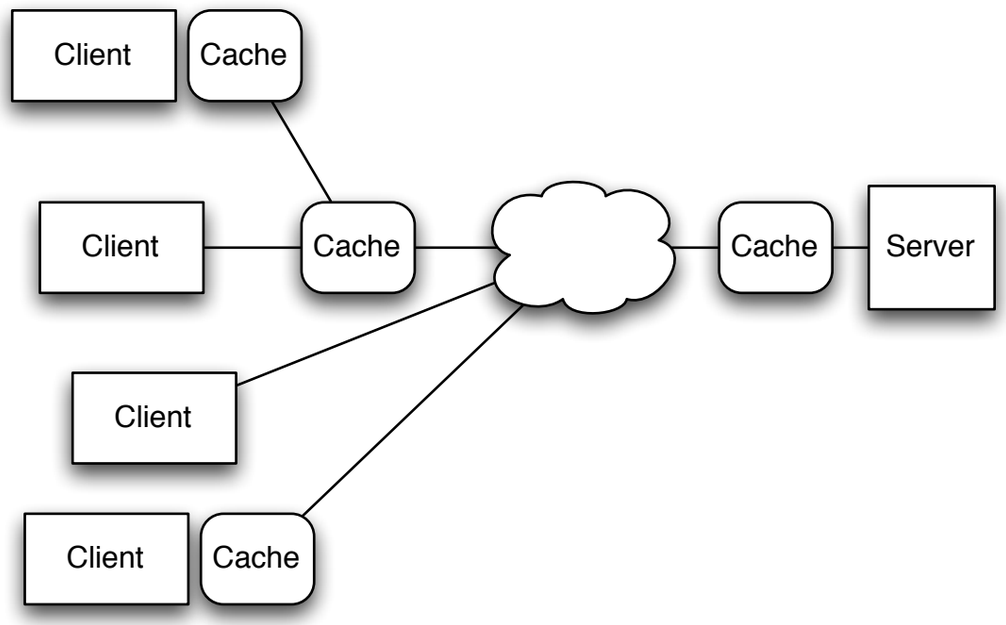
Client only



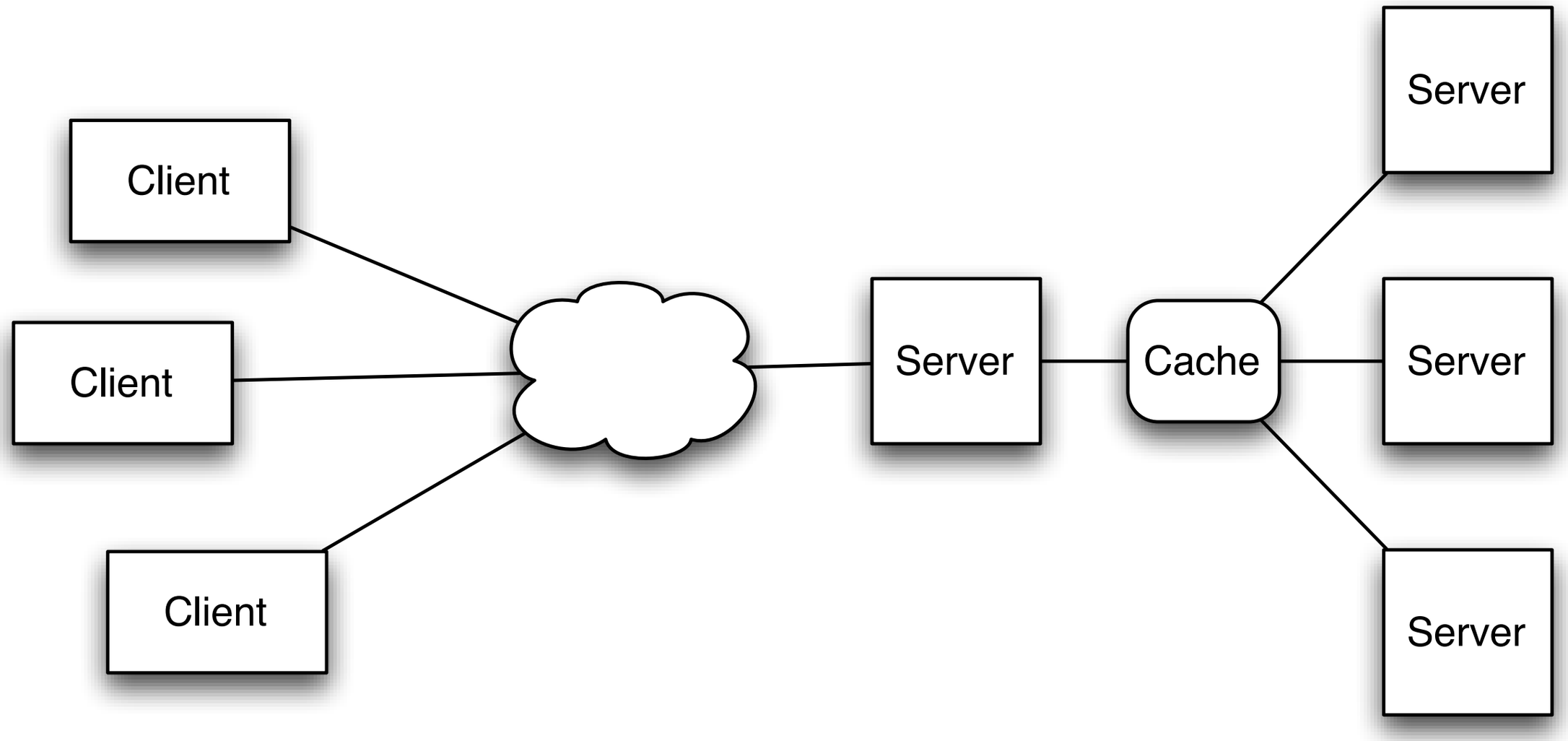
Proxy Cache



Reverse Proxy Cache



Complex Topology

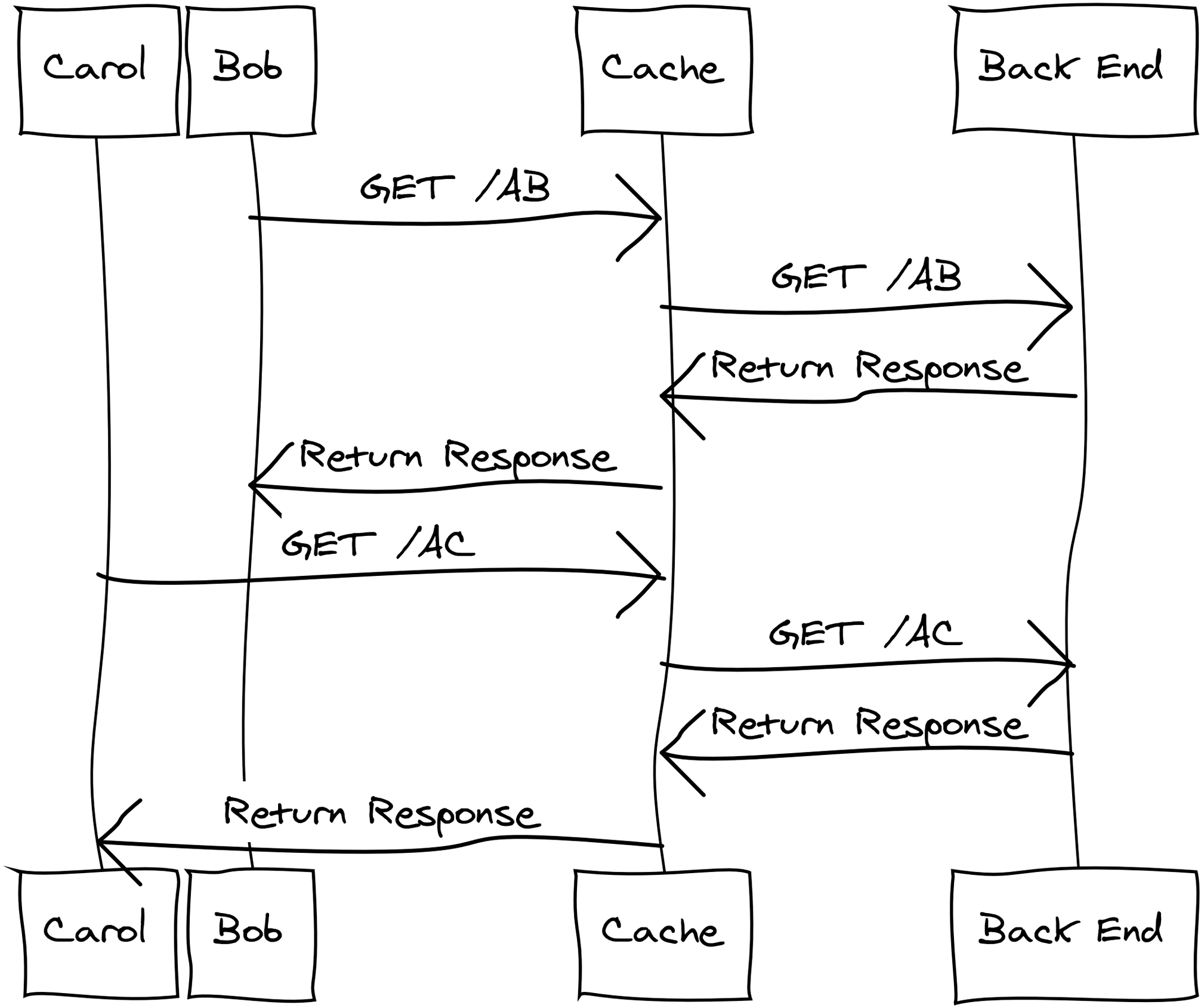


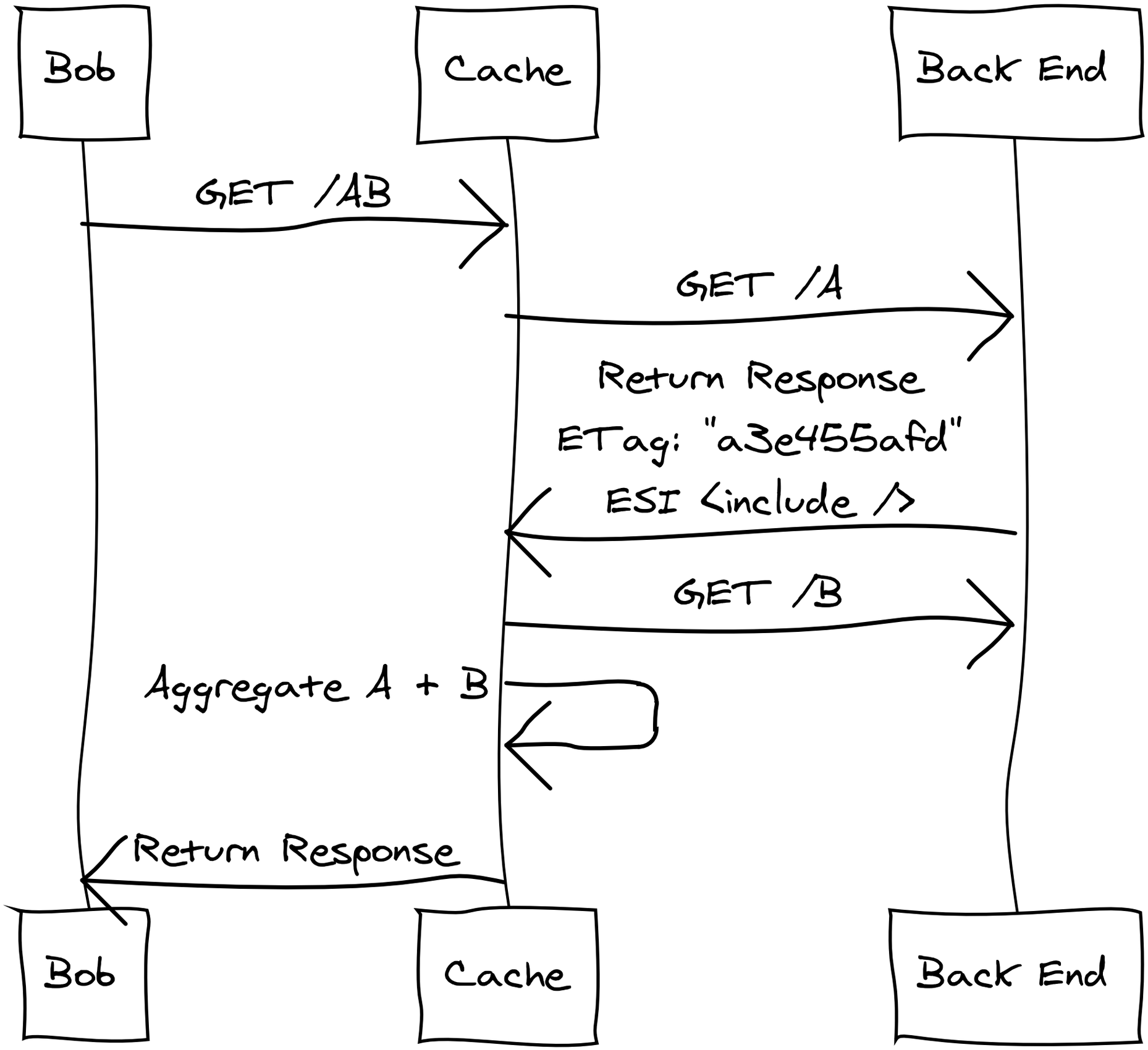
"Cache as ESB"

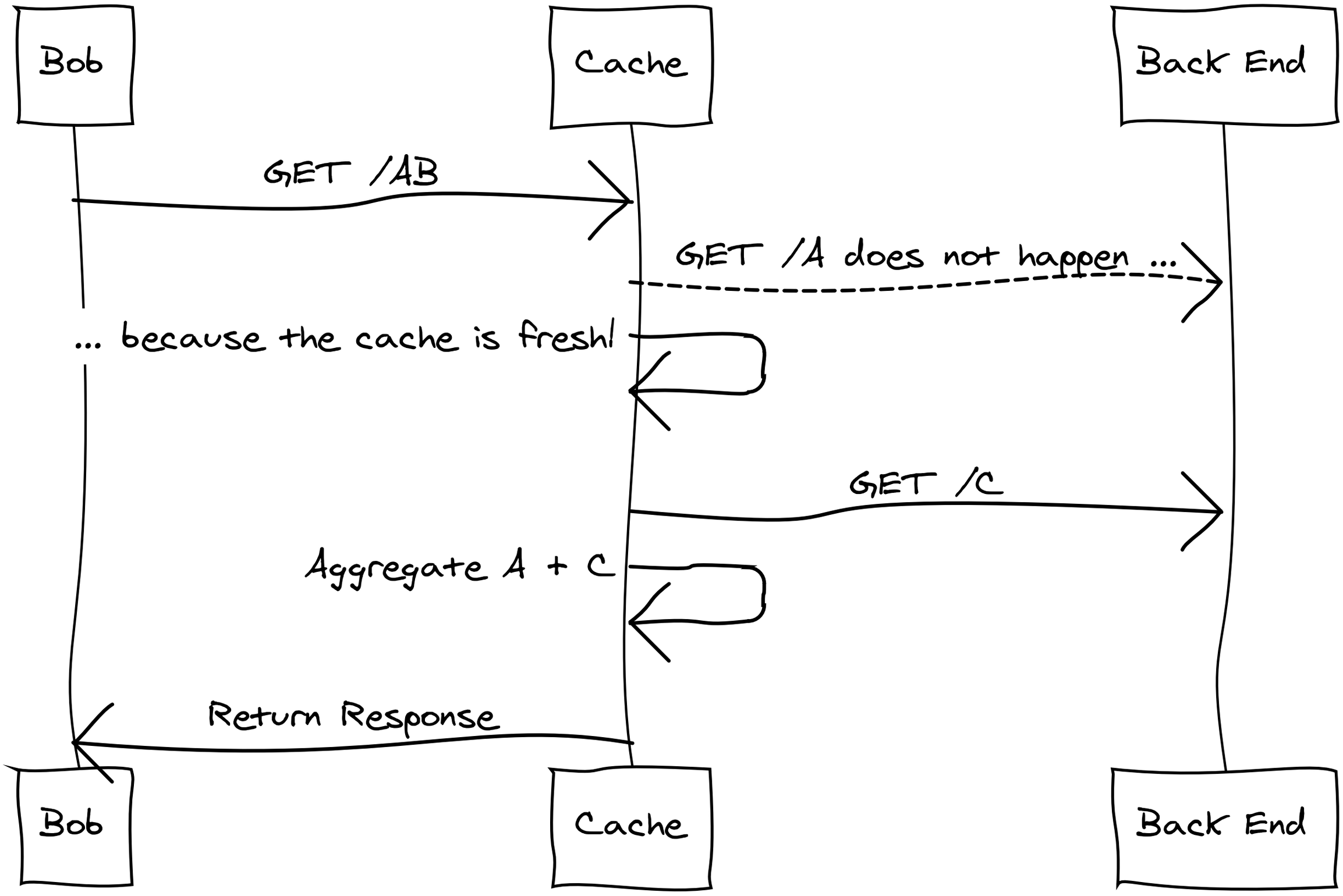
Edge Side Includes (ESI)

```
<esi:include
    src="http://example.com/1.html"
    alt="http://bak.example.com/2.html"
    onerror="continue"/>

<esi:include
    src="http://example.com/search?query=$(QUERY_STRING{query})"/>
```


Carol
Bob
Cache
Back End
GET /AB
GET /AB
Return Response
Return Response
GET /AC
GET /AC
Return Response
Return Response
Carol
Bob
Cache
Back End


Bob
Cache
Back End
GET /AB
GET /A
Return Response
ETag: "a3e455afd"
ESI <include />
GET /B
Aggregate A + B
Return Response
Bob
Cache
Back End


Bob
Cache
Back End
GET /AB
GET /A does not happen ...
... because the cache is fresh!
GET /C
Aggregate A + C
Return Response
Bob
Cache
Back End

Intermediaries

Squid

Full proxy cache w/ reverse proxy option

Mature/stable/old, widely used

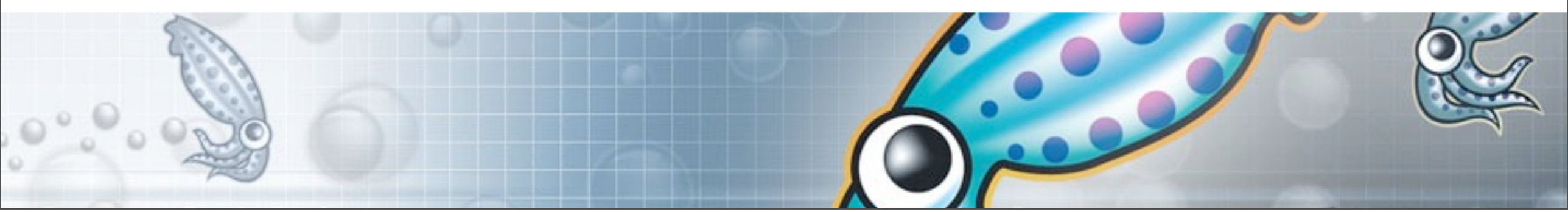
Complicated configuration

Support for ESI

Support for external invalidation (PURGE)

(Experimental) support for cache channels

(much, much more)



mod_cache

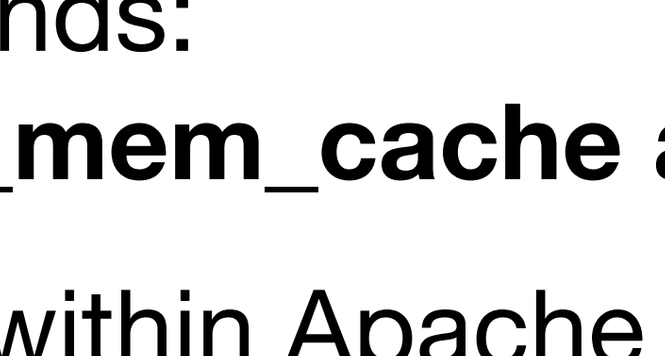
mod_cache module for Apache HTTPD 2.x (production-ready in 2.2)

backends:
mod_mem_cache and **mod_disk_cache**

Runs within Apache process

Simple solution in conjunction with Passenger

No support for ESI or explicit invalidation



```
<VirtualHost 1.2.3.4>
  ServerName example.com
  <Location /images>
    ExpiresActive On
    ExpiresDefault A3600
  </Location>
  <Location /user>
    ExpiresActive On
    ExpiresDefault "access plus 1 month"
  </Location>
  CacheEnable disk /images
  CacheRoot /var/www/cache
  ProxyPass / http://localhost:3000
  ProxyPassReverse / http://localhost:3000
</VirtualHost>
```

Varnish

Pure in-memory reverse proxy

Disk storage through OS swap mechanism

(Partial) ESI support

External invalidation

VCL configuration language

Easy configuration

```
sub vcl_recv {
    if (req.request != "GET" &&
      req.request != "HEAD" &&
      req.request != "PUT" &&
      req.request != "POST" &&
      req.request != "TRACE" &&
      req.request != "OPTIONS" &&
      req.request != "DELETE") {
        /* Non-RFC2616 or CONNECT
           which is weird. */
        return (pipe);
    }
    if (req.request != "GET"
        && req.request != "HEAD") {
        /* We only deal with GET and
           HEAD by default */
        return (pass);
    }
    if (req.http.Authorization
        || req.http.Cookie) {
        /* Not cacheable by default */
        return (pass);
    }
    return (lookup);
}
```

VCL

Hook	Meaning
vcl_recv	Called at the beginning of a request
vcl_pipe	Called upon entering pipe mode
vcl_pass	Called upon entering pass mode
vcl_hit	Called after a cache lookup if the requested document was found in the cache
vcl_miss	Called after a cache lookup if the requested document was not found in the cache
vcl_fetch	Called after a document has been successfully retrieved from the backend
vcl_deliver	Called before a cached object is delivered to the client
vcl_timeout	Called by the reaper thread shortly before a cached document reaches its expiry time
vcl_discard	Called by the reaper thread when a cached document is about to be discarded

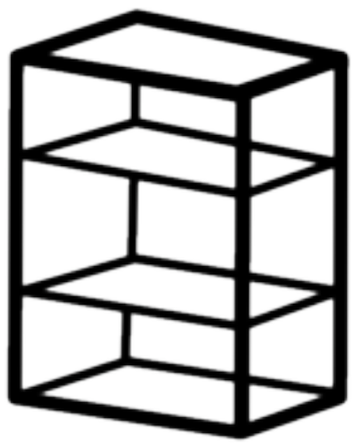
Rack::Cache

Runs within Ruby process

Cross-process synchronization via memcached

Absurdly simple + transparent

```
config.middleware.use(Rack::Cache,
    :verbose => true,
    :metastore   => 'file:/var/cache/rack/meta',
    :entitystore => 'file:/var/cache/rack/body')
```



rack::cache

powers web applications

REST & Rails

Rails < 2.0

```
ActionController::Routing::Routes.draw do |map|
  # ...
  map.connect ':controller/:action'
end
```

http://localhost:3000/demo/read_something?value1=...&value2=...

```
class DemoController < ApplicationController

  def read_something
    # retrieve some result using params[:value1], params[:value2], ...
  end

  def change_something
    # update backend using params[:value1], params[:value2], ...
  end
end
```

Rails < 2.0

Default (incl. scaffolding) unRESTful

URIs identify *actions*

No difference between POST and GET by default

Typical PHP/Java Web programming model

Rails ≥ 2.0

RESTful scaffolding as default

REST considered best practice

Uniform resource interface mapped to standard-style controllers

Routing

Simple:

```
map.resources :orders
```

Nested:

```
map.resources :customers do |customers|
  customers.resources :orders
end
```

Prefixed:

```
map.resources :comments, :path_prefix => '/articles/:article_id'
```

Scaffolding

```
$ script/generate scaffold Order customer_id:integer description:string
      exists  app/models/
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/orders
      exists  app/views/layouts/
      exists  test/functional/
      exists  test/unit/
      create  app/views/orders/index.html.erb
      create  app/views/orders/show.html.erb
      create  app/views/orders/new.html.erb
      create  app/views/orders/edit.html.erb
      create  app/views/layouts/orders.html.erb
   identical  public/stylesheets/scaffold.css
  dependency  model
      exists    app/models/
      exists    test/unit/
      exists    test/fixtures/
      create    app/models/order.rb
      create    test/unit/order_test.rb
      create    test/fixtures/orders.yml
      exists    db/migrate
      create    db/migrate/002_create_orders.rb
      create  app/controllers/orders_controller.rb
      create  test/functional/orders_controller_test.rb
      create  app/helpers/orders_helper.rb
       route  map.resources :orders
```

Controller Code

```
class OrdersController < ApplicationController
  # GET /orders
  # GET /orders.xml
  def index
  end

  # GET /orders/1
  # GET /orders/1.xml
  def show
  end

  # GET /orders/new
  # GET /orders/new.xml
  def new
  end

  # GET /orders/1/edit
  def edit
  end

  # POST /orders
  # POST /orders.xml
  def create
  end

  # PUT /orders/1
  # PUT /orders/1.xml
  def update
  end

  # DELETE /orders/1
  # DELETE /orders/1.xml
  def destroy
  end
end
```

Generated Routes

Output of rake routes:

```
    orders GET     /orders(.:format)              {:controller=>"orders", :action=>"index"}
           POST    /orders(.:format)             {:controller=>"orders", :action=>"create"}
 new_order GET     /orders/new(.:format)           {:controller=>"orders", :action=>"new"}
edit_order GET     /orders/:id/edit(.:format)     {:controller=>"orders", :action=>"edit"}
     order GET     /orders/:id(.:format)           {:controller=>"orders", :action=>"show"}
           PUT     /orders/:id(.:format)         {:controller=>"orders", :action=>"update"}
           DELETE  /orders/:id(.:format)          {:controller=>"orders", :action=>"destroy"}
                /:controller/:action/:id
                /:controller/:action/:id(.:format)
```

Content Negotiation

Deactivated by default (as of Rails 2.3)

Extension mapping instead

Activate through config:
action_controller.use_accept_header = true

Code via respond_to for responses,
manually for requests

Caching

Conditional GET in Rails

stale?	Use for handling failed validation explicitly
fresh_when	Use when rendering default template
request.fresh?	low-level access

Cache Options

Squid

Full proxy cache w/ reverse proxy option

Mature/stable/old, widely used

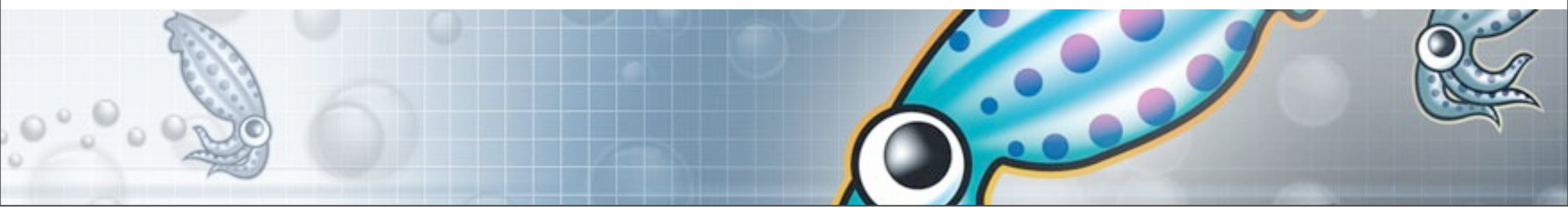
Complicated configuration

Support for ESI

Support for external invalidation

(Experimental) support for cache channels

(much, much more)



mod_cache

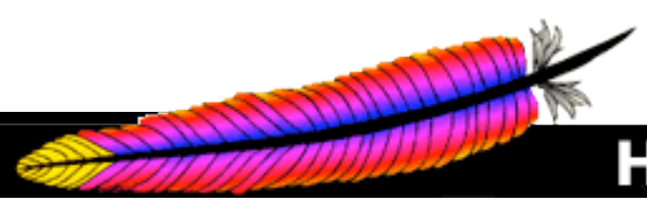
mod_cache module for Apache HTTPD 2.x (production-ready in 2.2)

backends:
mod_mem_cache and **mod_disk_cache**

Runs within Apache process

Simple solution in conjunction with Passenger

No support for ESI or explicit invalidation



```
<VirtualHost 1.2.3.4>
  ServerName example.com
  <Location /images>
    ExpiresActive On
    ExpiresDefault A3600
  </Location>
  <Location /user>
    ExpiresActive On
    ExpiresDefault "access plus 1 month"
  </Location>
  CacheEnable disk /images
  CacheRoot /var/www/cache
  ProxyPass / http://localhost:3000
  ProxyPassReverse / http://localhost:3000
</VirtualHost>
```

Varnish

Pure in-memory reverse proxy

Disk storage through OS swap mechanism

(Partial) ESI support

VCL configuration language

Easy configuration

VCL

```
sub vcl_recv {
   if (req.request != "GET" &&
     req.request != "HEAD" &&
     req.request != "PUT" &&
     req.request != "POST" &&
     req.request != "TRACE" &&
     req.request != "OPTIONS" &&
     req.request != "DELETE") {
       /* Non-RFC2616 or CONNECT
          which is weird. */
       return (pipe);
   }
   if (req.request != "GET"
       && req.request != "HEAD") {
       /* We only deal with GET and
          HEAD by default */
       return (pass);
   }
   if (req.http.Authorization
       || req.http.Cookie) {
       /* Not cacheable by default */
       return (pass);
   }
   return (lookup);
}
```

Hook	Meaning
vcl_recv	Called at the beginning of a request
vcl_pipe	Called upon entering pipe mode
vcl_pass	Called upon entering pass mode
vcl_hit	Called after a cache lookup if the requested document was found in the cache
vcl_miss	Called after a cache lookup if the requested document was not found in the cache
vcl_fetch	Called after a document has been successfully retrieved from the backend
vcl_deliver	Called before a cached object is delivered to the client
vcl_timeout	Called by the reaper thread shortly before a cached document reaches its expiry time
vcl_discard	Called by the reaper thread when a cached document is about to be discarded

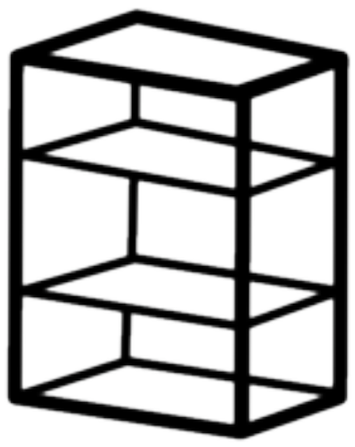
Rack::Cache

Runs within Ruby process

Cross-process synchronization via memcached

Absurdly simple + transparent

```
config.middleware.use(Rack::Cache,
  :verbose => true,
  :metastore   => 'file:/var/cache/rack/meta',
  :entitystore => 'file:/var/cache/rack/body')
```



rack::cache

powers web applications

Conditional PUT

Use If-Match/If-Not-Modified-Since

```
def update
  @customer = Customer.find(params[:id])
  if precondition_met?(@customer)
    respond_to do |format|
      if @customer.update_attributes_from_xml(params[:customer])
        format.xml  {
          response.content_type = 'application/vnd.innoq-customer+xml'
          render :action => 'show'
        }
        else
          format.xml  { render :xml => @customer.errors,
                               :status => :unprocessable_entity }
        end
      end
    else
      head :precondition_failed
    end
  end

  def precondition_met?(obj)
    response.etag = obj
    if_match_header = request.headers['IF-MATCH']
    if_match_header.nil? || (if_match_header == response.etag)
  end
```

Atom Syndication

```
xml.instruct! :xml, :encoding => "UTF-8"
xml.feed :xmlns => 'http://www.w3.org/2005/Atom', 'xml:base' => base_uri, 'xml:lang' => 'en-us' do
  xml.link :rel => 'self', :type => 'application/atom+xml', :href => request.url
  xml.link :rel => 'alternate', :type => 'text/html', :href => orders_url
  will_paginate(@orders, :xml => true, :builder => xml)
  xml.id orders_url
  xml.title 'Feed for OM order updates'
  xml.updated
  for order in @orders do
    xml.entry do |entry|
      entry.id order_url(order)
      entry.published order.created_at.utc
      entry.updated order.updated_at.utc
      entry.link :href => order_path(order), :type => 'text/html', :rel => 'alternate'
      entry.title "Order from #{order.customer_description}, amount #{order.total}"
      entry.summary :type => 'xhtml' do |xhtml|
        xhtml.div :xmlns => 'http://www.w3.org/1999/xhtml'
        xhtml << render(:partial => 'order', :locals => { :order => order })
      end
      entry.author do |author|
        author.name "OM System"
        author.email "orders@om.example.com"
      end
    end
  end
end
```

Clients

ActiveResource:

- Client for RESTful Rails Resources
- Basic support for ActiveRecord-style code

```
class Person < ActiveResource::Base
  self.site = "http://api.people.com:3000/"
end

# Find a person with id = 1
ryan = Person.find(1)
Person.exists?(1)  #=> true
```

Clients

REST-Client

- http://rest-client.heroku.com
- Consume *any* RESTful service

```
require 'rest_client'

RestClient.get 'http://example.com/resource'
RestClient.get 'https://user:password@example.com/private/resource'

RestClient.post 'http://example.com/resource',
              :param1 => 'one', :nested => { :param2 => 'two' }
RestClient.delete 'http://example.com/resource
```

RESTful APIs

RESTful APIs don’t expose low-level details

Same layer – different abstraction

Value through uniformity and hypermedia

Mapping necessity: “Implement” HTTP base interface

How RESTful is Rails?

Positive:

Consistent and clean CRUD mapping

Use of URIs for resource identification

Support for content negotiation

Reasonable Status codes

ETags (!)

How RESTful is Rails?

Negative:

No hypermedia

CRUD-centric

Proprietary protocol for ActiveResource

My Rails/REST Wishlist

A really cool, meta-driven hypermedia programming model

for both client and server (w/o coupling)

(Better) Atom Syndication and Atom Pub Support