

Effective Network Programming with iOS 7

Ben Scheirman

- Director of Development, ChaiONE
- Houston, TX
- Guitar Player
- Beer Geek

benscheirman.com

@subdigital

ChaiOne 



Bite-sized Screencasts

NS*Screencast*.com

Weekly videos on iOS Development

\$9/month

Why is this important?

99% of our apps
are not self-contained

Why is this important?

Doing it wrong
RUINS user experience

Why is this important?

Too many apps do it wrong

Exhibit A: Stuttery Scroller

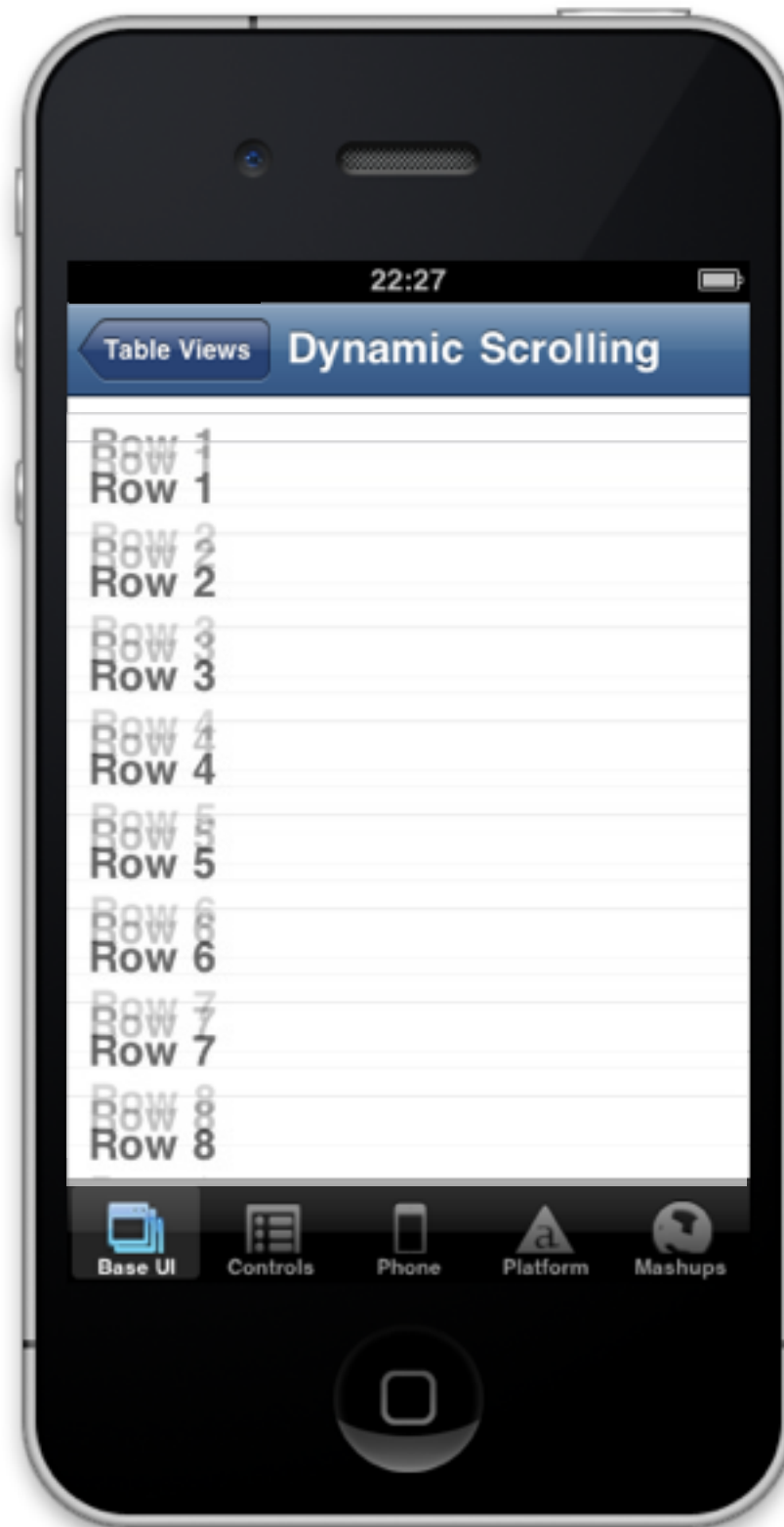
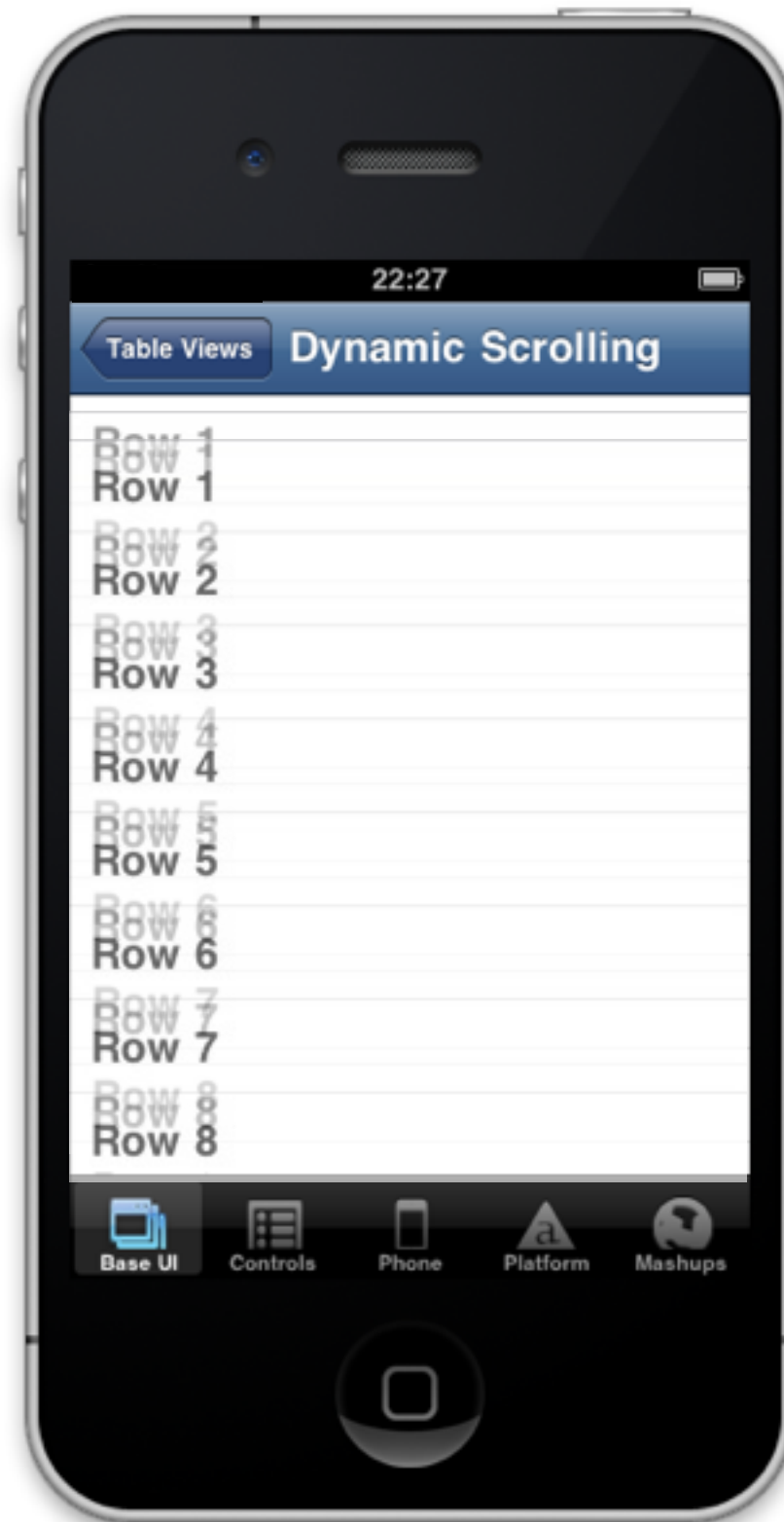


Exhibit A: Stuttery Scroller



Accessing the
network on the
main thread?

Exhibit B: The Perpetual Loader

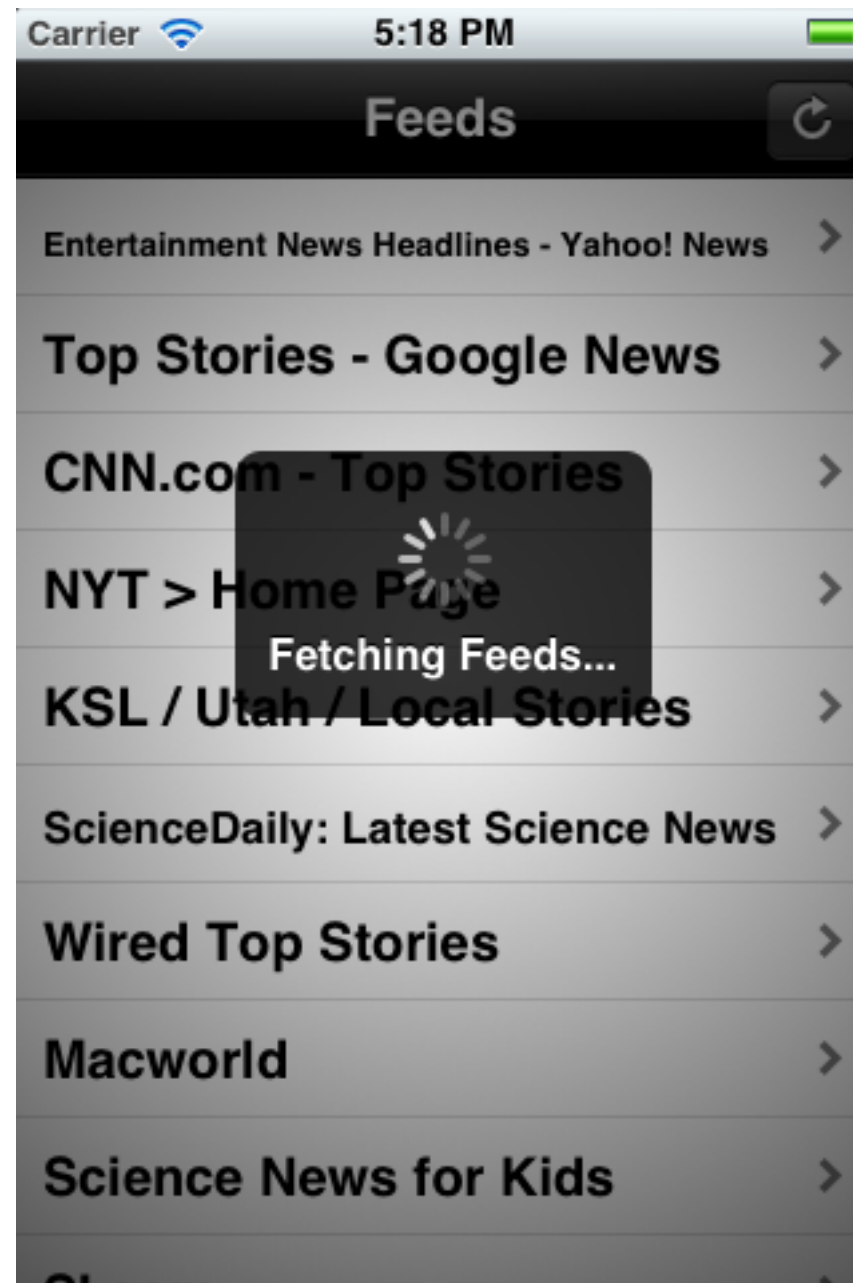
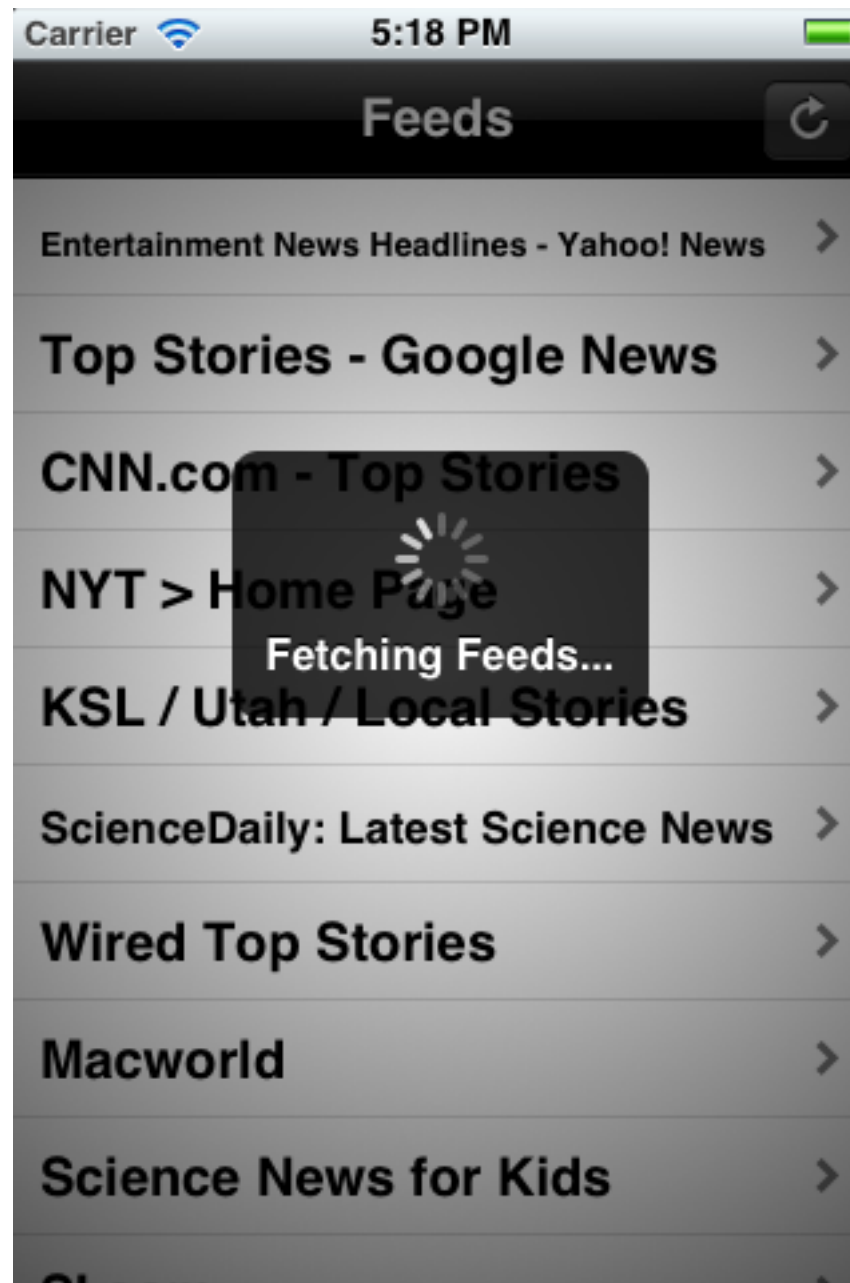


Exhibit B: The Perpetual Loader



How about
you cache
some data?

Exhibit C: Slow Loader

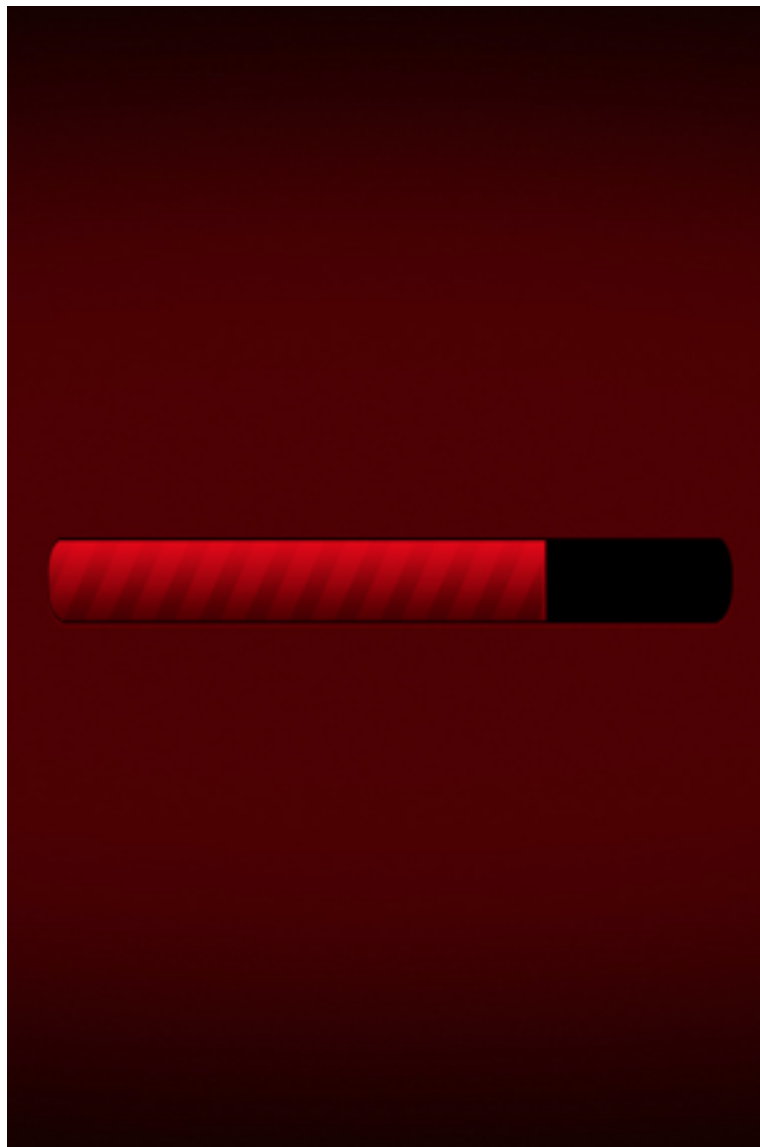
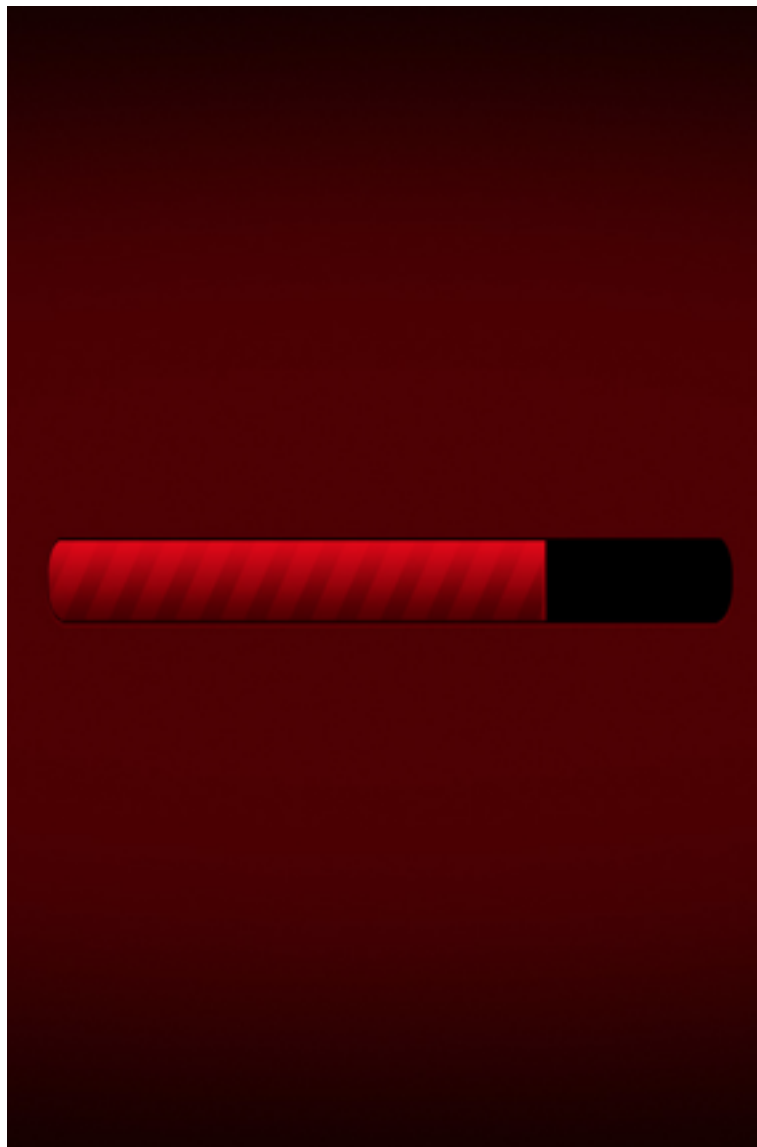


Exhibit C: Slow Loader



Loading too
much data?

Exhibit D: Yesterday's News

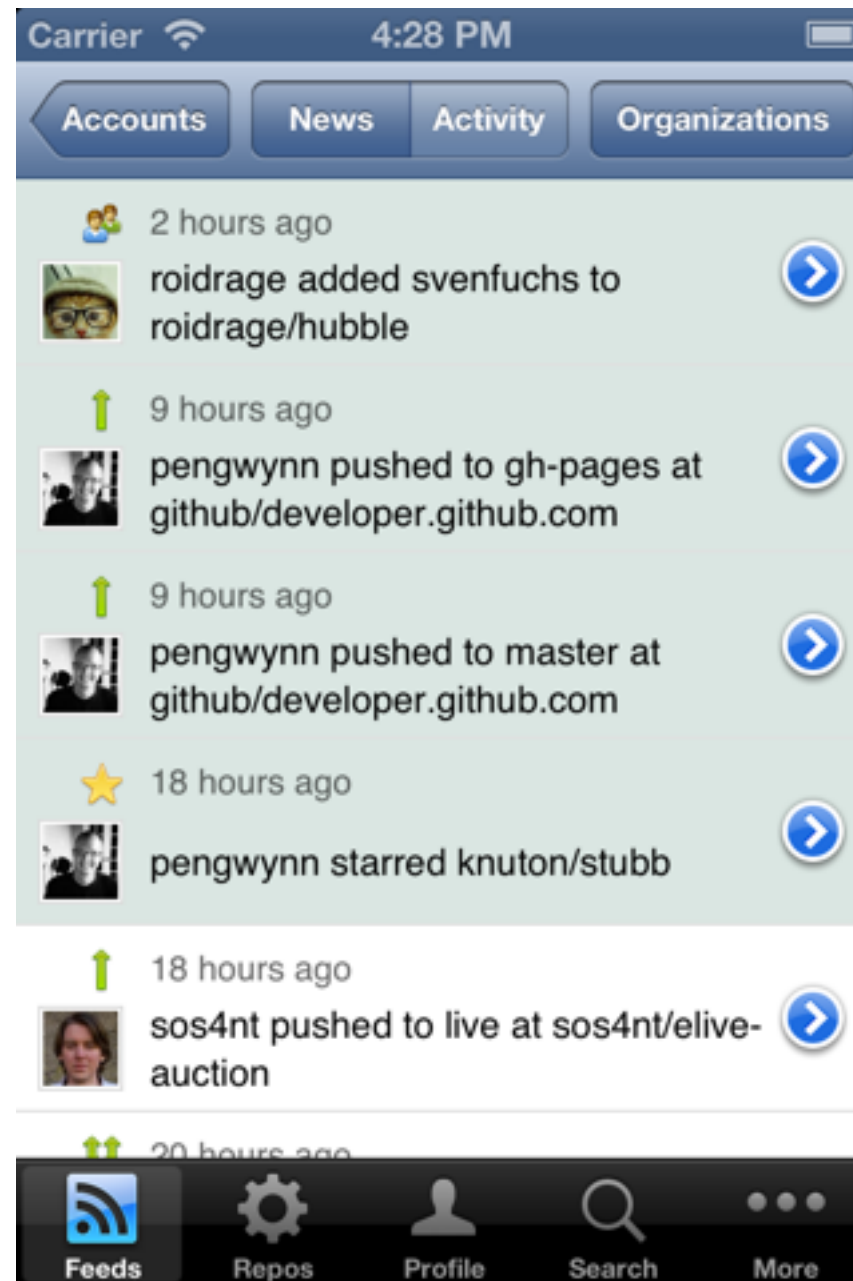
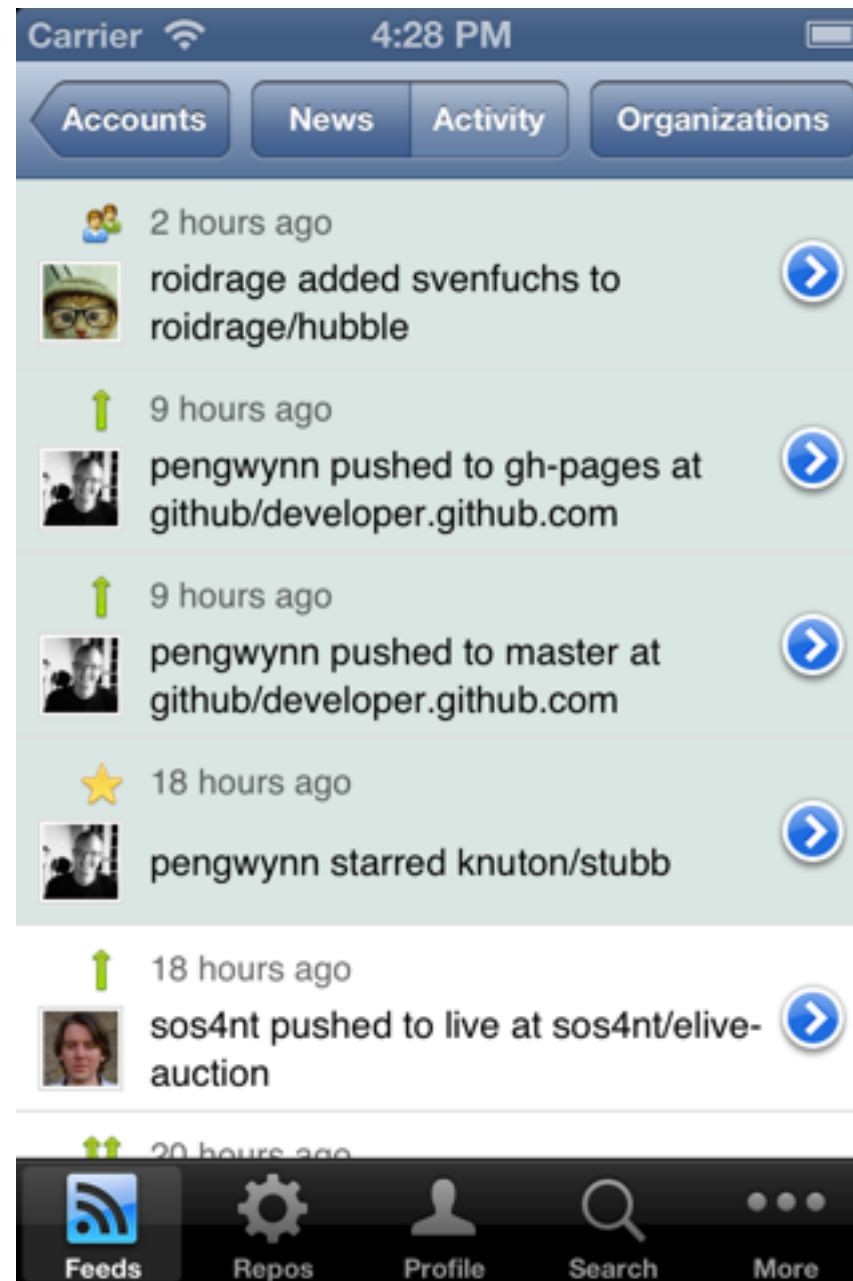


Exhibit D: Yesterday's News



Caching for
too long?

What we'll cover

NSURLSession Basics

LIVE CODING (wish me luck!)

AF Networking 2.0

HTTP Caching

Caching Patterns

API Design Tips

NSURLConnection

**Invented for Safari ~2000, made
public in 2003**

**Poor separation of settings,
config, cache**

**Now replaced by NSURLSession
classes**

NSURLConnection

Invented for Safari ~2000, made public in 2003

Poor separation of settings, config, cache

Now replaced by NSURLSession classes



NSURLSession

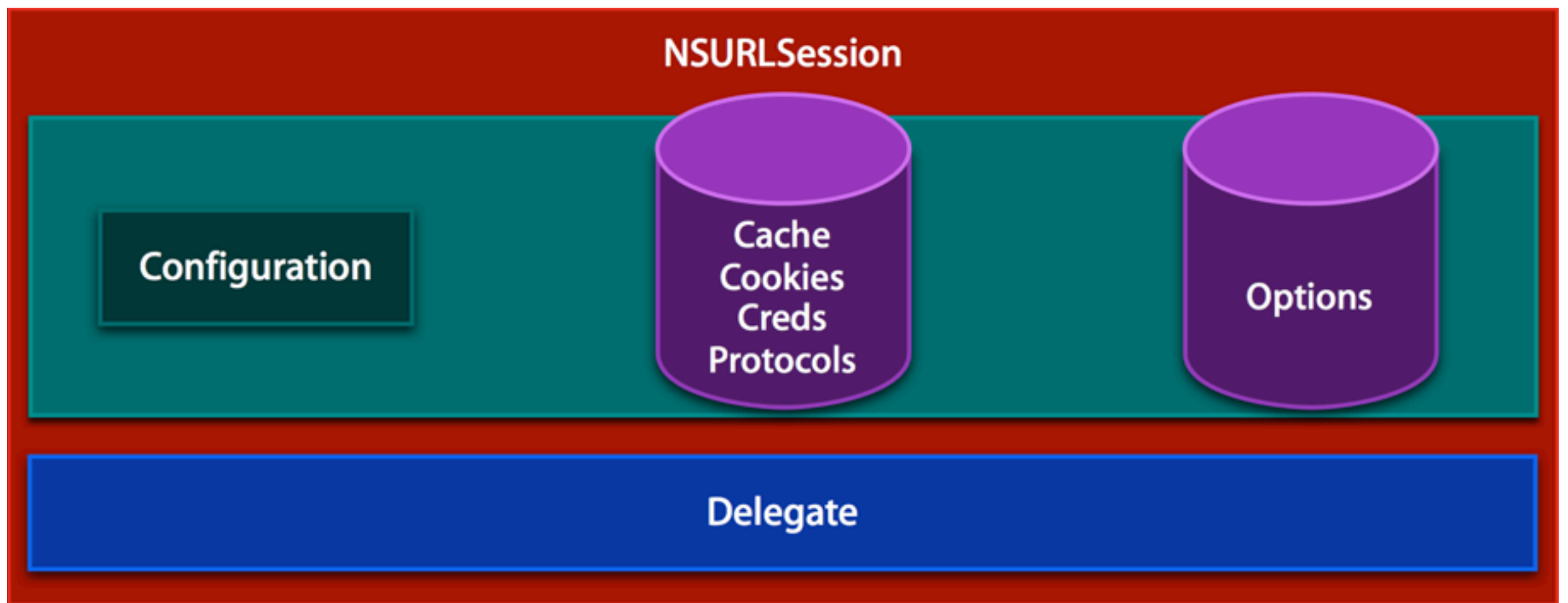
Still use NSURLRequest

Configurable Container (isolation w/ 3rd party libraries)

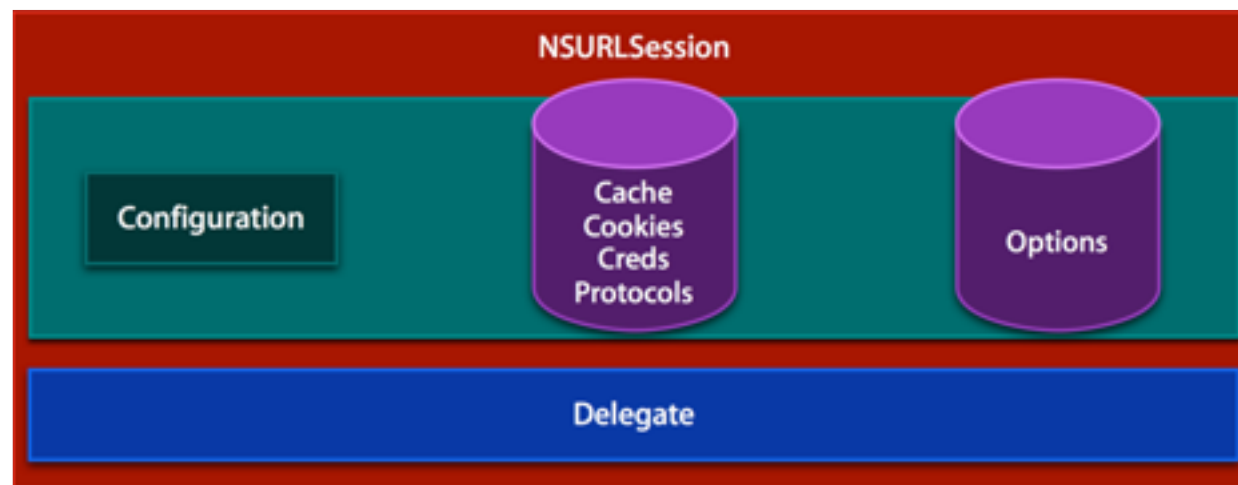
Improved auth

Rich Callbacks

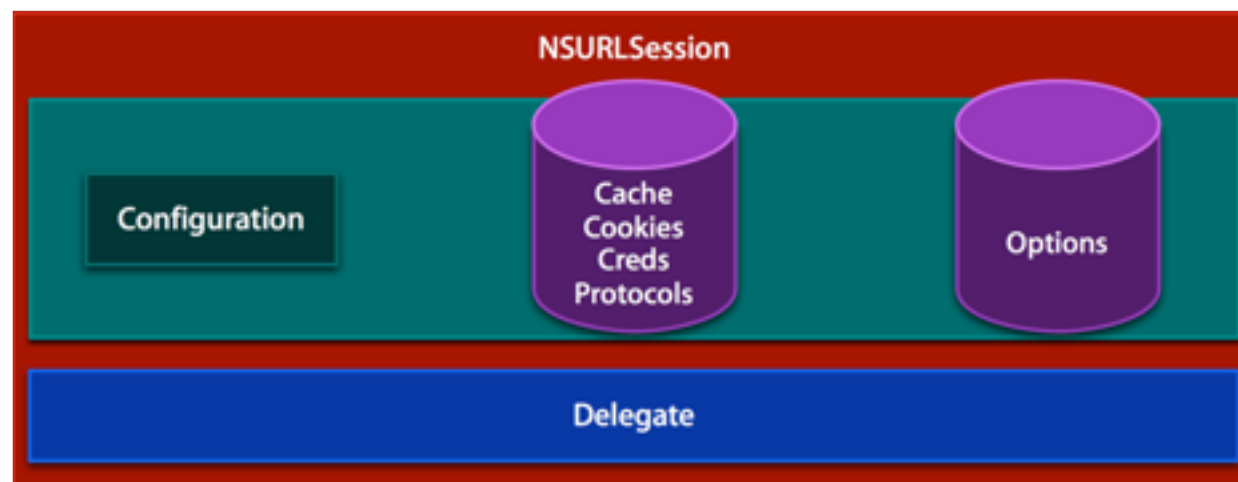
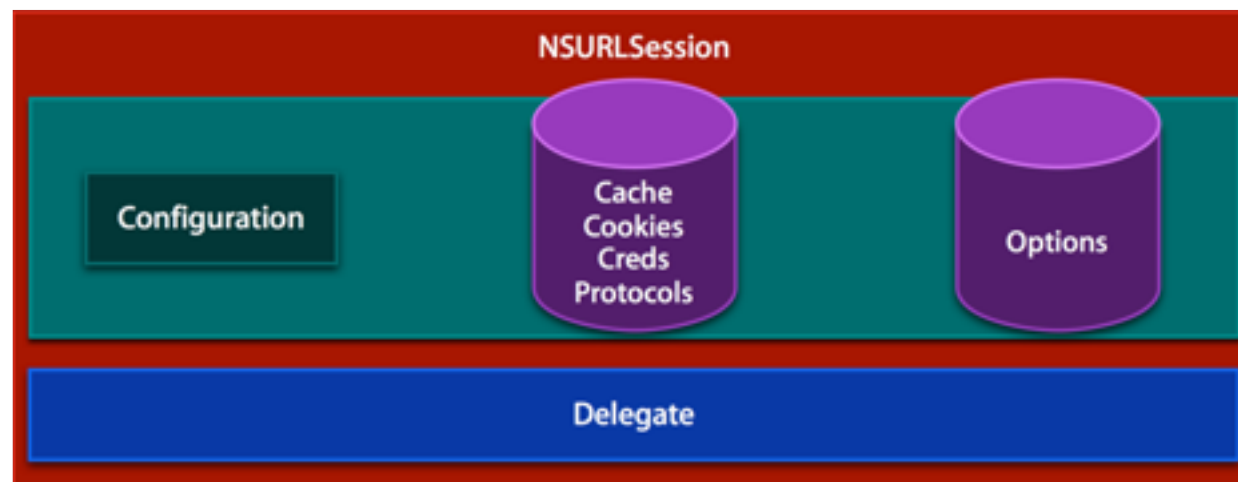
NSURLSession



NSURLSession



NSURLSession



NSURLSession Class Family

NSURLSession

NSURLSessionConfiguration

NSURLSessionTask

NSURLSessionDelegate

NSURLSession

NSURLSession

Interface for issuing network calls against a server

Likely use case: 1 session per API

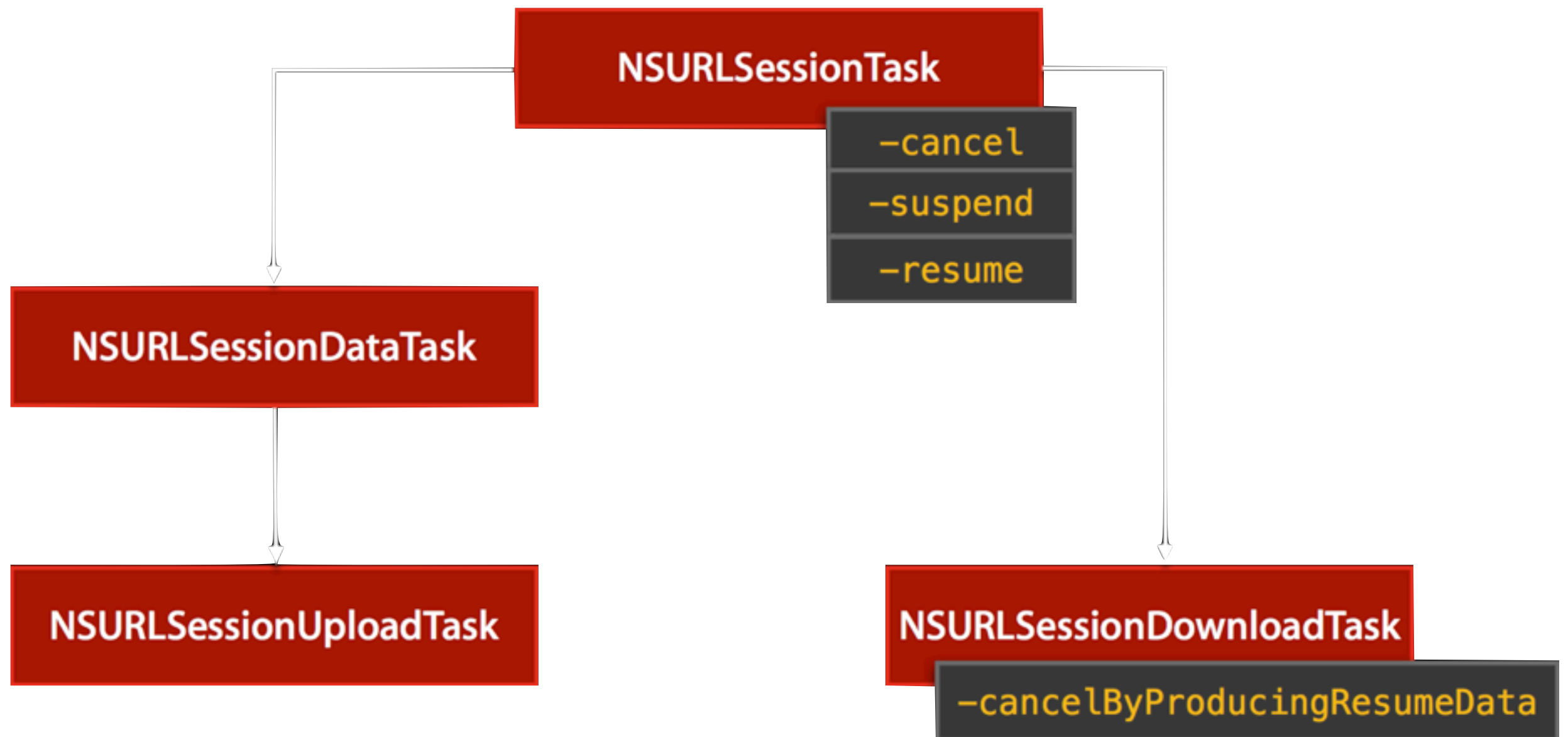
NSURLSessionDelegate

Delegate methods for a session:

- `NSURLSession:didBecomeInvalidWithError:`
- `NSURLSession:didReceiveChallenge:completionHandler:`

Used for connect level authentication challenges
NTLM, Kerberos, Client certificates

NSURLSessionTask



NSURLSessionTaskDelegate

Delegate methods for a request:

- `NSURLSession:task:willPerformHTTPRedirection:newRequest:completionHandler:`
- `NSURLSession:task:didReceiveChallenge:completionHandler:`
- `NSURLSession:task:needNewBodyStream:`
- `NSURLSession:task:didSendBodyData:totalBytesSent:totalBytesExpectedToSend:`
- `NSURLSession:task:didCompleteWithError:`

Only need to implement for fine-grained control

NSURLSessionDataTaskDelegate

Delegate methods for a data request:

- `NSURLSessionDataTask:didReceiveChallenge:completionHandler:`
- `NSURLSessionDataTask:didBecomeDownloadTask:`
- `NSURLSessionDataTask:didReceiveData:`
- `NSURLSessionDataTask:willCacheResponse:completionHandler:`

Only need to implement for fine-grained control

NSURLSessionDownloadDelegate

- `NSURLSession:downloadTask:didFinishDownloadingToURL:`
- `NSURLSession:downloadTask:didWriteData:totalBytesWritten:totalBytesExpectedToWrite:`
- `NSURLSession:downloadTask:didResumeAtOffset:expectedTotalBytes:`

Requesting Images

Have you ever seen this?

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/kwpjYwQ.jpg"];  
  
NSData *imageData = [NSData  
    dataWithContentsOfURL:imageUrl];  
  
UIImage *image = [UIImage imageData:imageData];  
  
[imageView setImage:image];
```

Have you ever seen this?

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/kwpjYwQ.jpg"];  
  
NSData *imageData = [NSData  
    dataWithContentsOfURL:imageUrl];  
  
UIImage *image = [UIImage imageData:imageData];  
[imageView setImage:image];
```

WRONG

Have you ever seen this?

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/kwpjYwQ.jpg"];
```



```
NSData *imageData = [NSData  
    dataWithContentsOfURL:imageUrl];
```



```
UIImage *image = [UIImage imageData:imageData];  
[imageView setImage:image];
```

In case you were wondering...



MAY 93) One woman in 100 Mile House hospital, the other recovering in Vancouver

BY LAURA P. GUNN
STAFF WRITER

pair of Kluge's shoes have had to fill in for my mother's with me in the 100th.

month ago, Thomas Bradley, 49, badly injured when she hit a moose with her vehicle, was in Vancouver recovering from her injuries.

Cape Mudge, British Columbia. One woman in 100 Mile House hospital, the other recovering in Vancouver.

In the first accident, a moose hit Bradley on her way home from a business trip when a moose ran in front of her car.

The accident went through the windshield and landed on her, breaking a broken wrist, arm and hand, five fractured ribs and bleeding in her brain.

The pregnant moose died near the collision.

The calf did not survive.

When Bradley came out of her

coma last week, Bradley and her husband 50-year-old decided to stay here.

She and her husband were in two cars last Friday afternoon. In the first car, she was going around a curve near the 100 Mile House when she saw "a brown blur."

"It was right in front of me," she said. "I slammed on the brakes with both my feet."

"It was like two explosions. [The moose] came through the windshield on the passenger side, and then the body flipped around and the shoulder came inside the window."

She was taken to the hospital in 100 Mile House with serious soft-tissue injuries.

The vehicle's airbags were just like her sister "never knew me for a long."

"My first thought was, 'Are the women going out (in a) hunting season for my family?'" Bradley said. "So far, we win three because we got three of them dead."

Moose are involved in about eight per cent of all wildlife vehicle collisions, according to the Wildlife Collision Prevention Program's website.

"Moose will often try to avoid vehicles by running along a highway," said Jeff Knight, spokesman for the Ministry of Environment.

"It's really important to stay alert when you drive through the forest."

Bradley was released from hospital Sunday, and Monday she was home, she said and was before.

"You can never tell when a moose or some other wildlife will show up in front of you," she said. "I've never almost died from one. It's worse when you're alone, that's how I died."

Bradley is the person

Can on tailgating may be lifted

Demo

- **Demo: Download an image with
NSURLSessionDownloadTask**

NSURLSessionDataTask

- Represents GET/PUT/POST/DELETE
- Handle completion callback
- If `error` is present, request failed

What constitutes an error?

- **Connection failed**
- **Timeouts**
- **Host invalid**
- **Bad URL**
- **Too many redirects**
- **... dozens more (check URL Loading System Error Codes)**

Demo

- **Demo: Searching the iTunes Store**

AFNetworking

AFNetworking 2.0



Installation

```
gem install cocoapods
```

Podfile

```
platform :ios, "7.0"
```

```
pod 'AFNetworking', '2.0'
```

```
pod install
```



Installation

```
gem install cocoapods
```

Podfile

```
platform :ios, "7.0"
```

```
pod 'AFNetworking', '2.0'
```

```
pod install
```



Installation

```
gem install cocoapods
```

Podfile

```
platform :ios, "7.0"  
  
pod 'AFNetworking', '2.0'
```

```
pod install
```



Installation

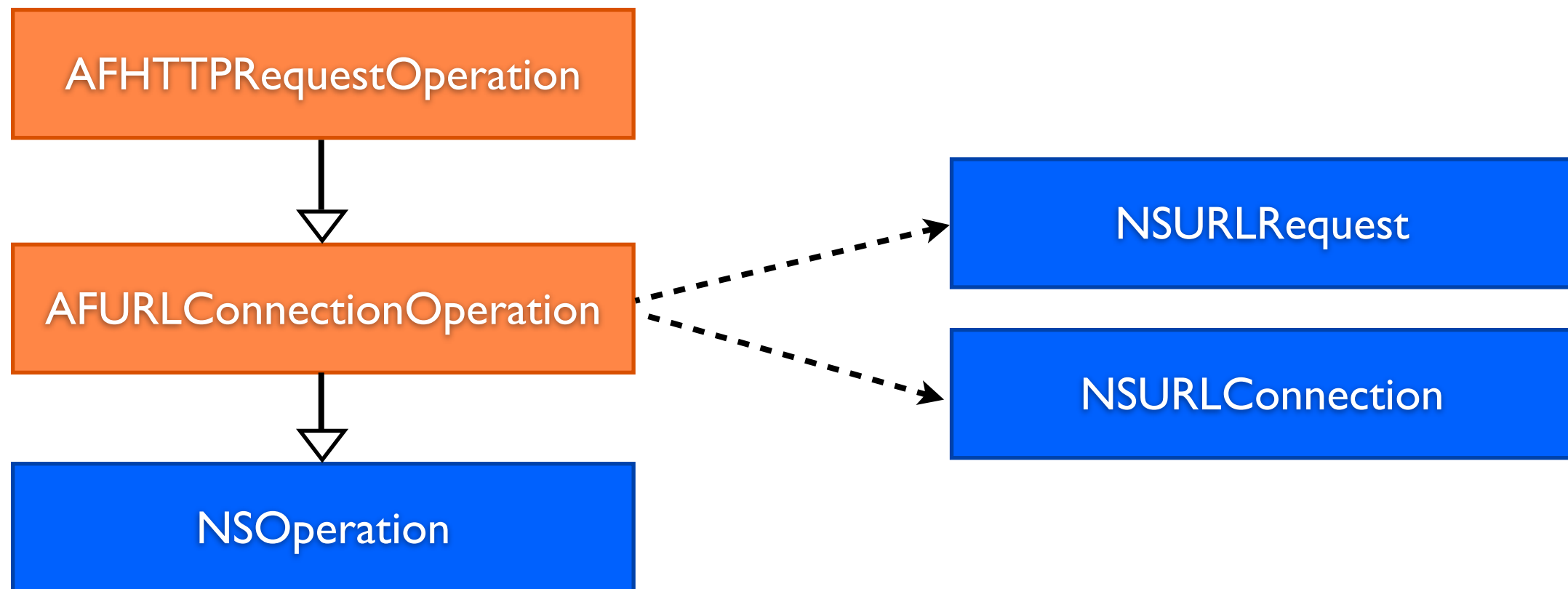
```
gem install cocoapods
```

Podfile

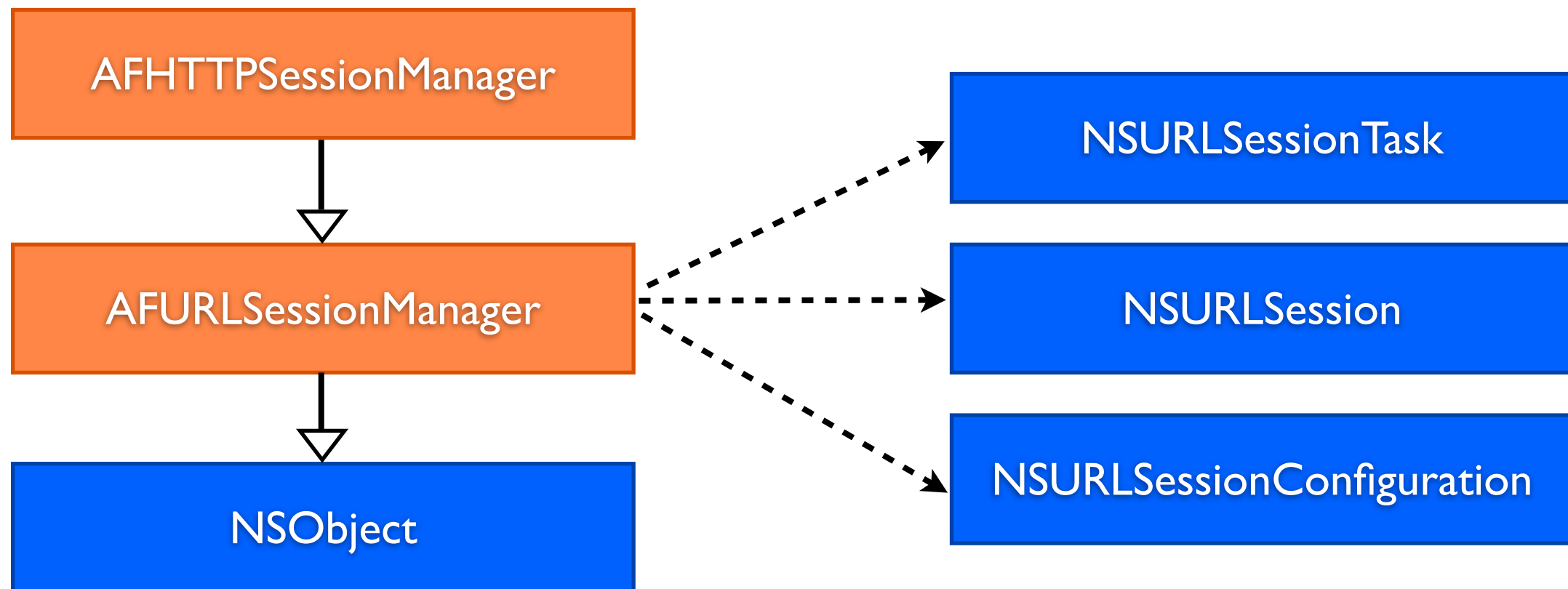
```
platform :ios, "7.0"  
  
pod 'AFNetworking', '2.0'
```

```
pod install
```

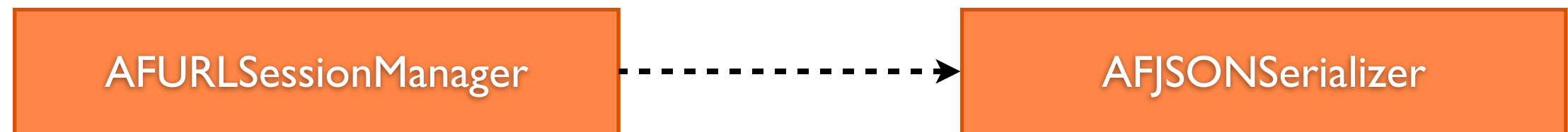
AFNetworking 1.0 Design



AFNetworking 2.0 Design



Serializers



Usage

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLSessionConfiguration *config = [NSURLSessionConfiguration
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];  
  
NSURLSessionConfiguration *config = [NSURLSessionConfiguration  
                                     defaultConfiguration];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];  
  
NSURLSessionConfiguration *config = [NSURLSessionConfiguration  
                                     defaultConfiguration];  
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];

AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
manager.responseSerializer = [[AFJSONSerializer alloc] init];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
manager.responseSerializer = [[AFJSONSerializer alloc] init];
[manager GET:url parameters:nil completion:^(NSURLSessionDataTask *task,
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
manager.responseSerializer = [[AFJSONSerializer alloc] init];
[manager GET:url parameters:nil completion:^(NSURLSessionDataTask *task,
      id responseObject) {
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
manager.responseSerializer = [[AFJSONSerializer alloc] init];
[manager GET:url parameters:nil completion:^(NSURLSessionDataTask *task,
      id responseObject) {
    // handle success
}
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
manager.responseSerializer = [[AFJSONSerializer alloc] init];
[manager GET:url parameters:nil completion:^(NSURLSessionDataTask *task,
      id responseObject) {
    // handle success
} failure:^(NSURLSessionDataTask *task, NSError *error) {
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
manager.responseSerializer = [[AFJSONSerializer alloc] init];
[manager GET:url parameters:nil completion:^(NSURLSessionDataTask *task,
      id responseObject) {
    // handle success
} failure:^(NSURLSessionDataTask *task, NSError *error) {
    // handle failure
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                                     defaultConfiguration];
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc]
                                   initWithSessionConfiguration:config];
manager.responseSerializer = [[AFJSONSerializer alloc] init];
[manager GET:url parameters:nil completion:^(NSURLSessionDataTask *task,
      id responseObject) {
    // handle success
} failure:^(NSURLSessionDataTask *task, NSError *error) {
    // handle failure
}];
```

Requesting Images

Requesting Images

```
NSURL *imageUrl = [NSURL URLWithString:
```

Requesting Images

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/kwpjYwQ.jpg"];
```

Requesting Images

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/kwpjYwQ.jpg"];  
  
[imageView setImageWithURL:imageUrl];
```

Requesting XML?

Requesting XML?

AFXMLResponseSerializer

Requesting plist?

Requesting plist?

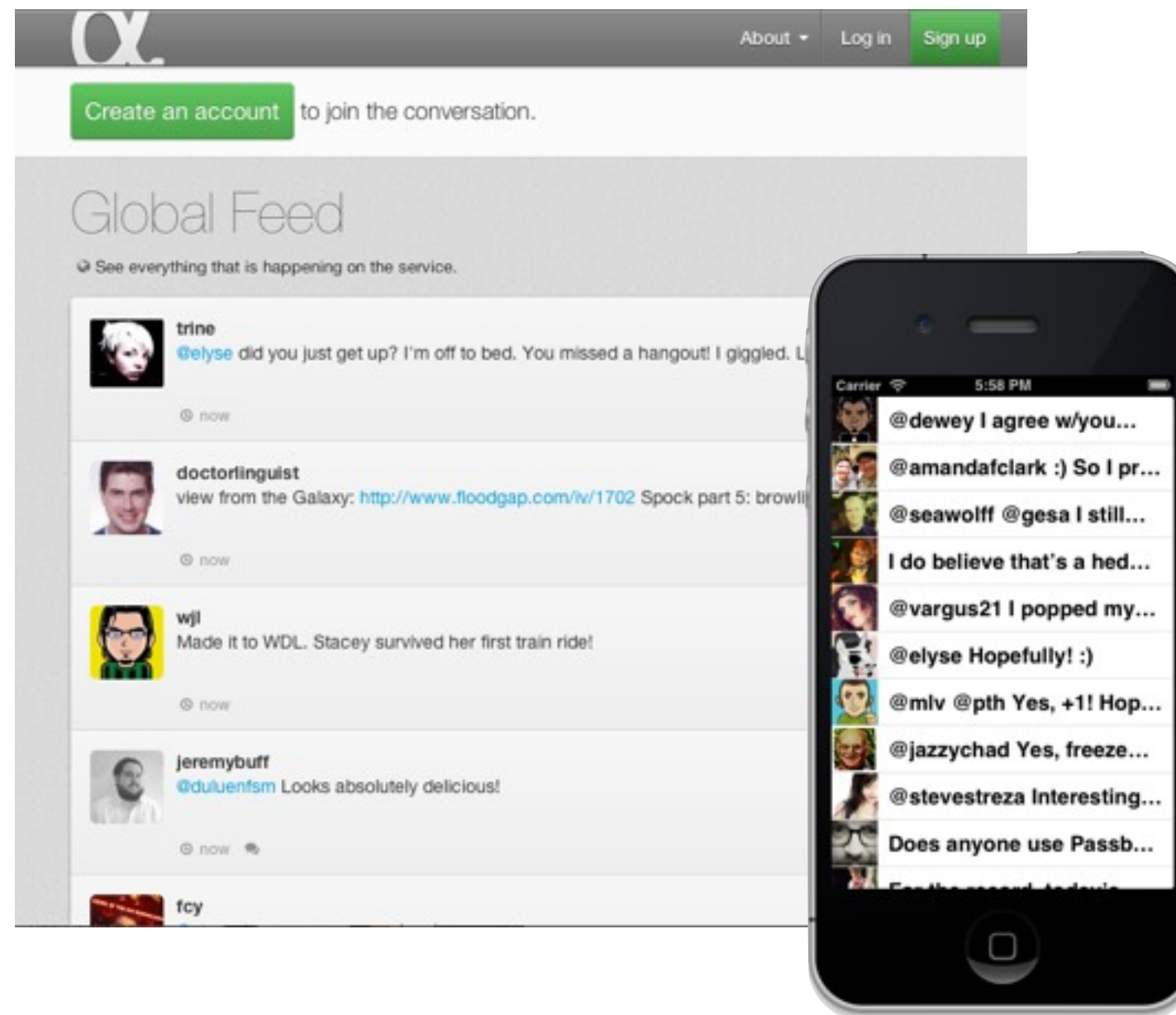
AFPPropertyListResponseSerializer

Requesting a custom data format?

Requesting a custom data format?

MyCustomResponseSerializer

Demo: app.net client



Other AFNetworking Goodies

AFHTTPSessionManager setSecurityPolicy

[AFSecurityPolicy policyWithPinningMode:AFSSLPinningModePublicKey]

Other AFNetworking Goodies

UIActivityIndicatorView+AFNetworking.h

```
[indicator setAnimatingWithStateOfTask:task]
```

HTTP Status Code Cheat Sheet

1xx - Not Used

2xx - Thumbs Up

3xx - It's over there

4xx - You messed up

5xx - We messed up

HTTP Caching

Client

Server

HTTP Caching

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com
```



HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



The diagram shows a grey rounded rectangle containing the request text. A thick grey arrow points from the right side of this rectangle towards the right, indicating the direction of the request.



The diagram shows a purple rounded rectangle containing the response text. A thick grey arrow points from the left side of this rectangle towards the left, indicating the direction of the response.

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

Leveraging Etag on the client

Client

Server

Leveraging Etag on the client

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-None-Match: "e6482391249374127ca94128"
```



Leveraging Etag on the client

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-None-Match: "e6482391249374127ca94128"
```



HTTP/1.1 304 Not Modified

< no response body >



Leveraging Modified Date

Client

Server

Leveraging Modified Date

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-Modified-Since: <date>
```



Leveraging Modified Date

Client

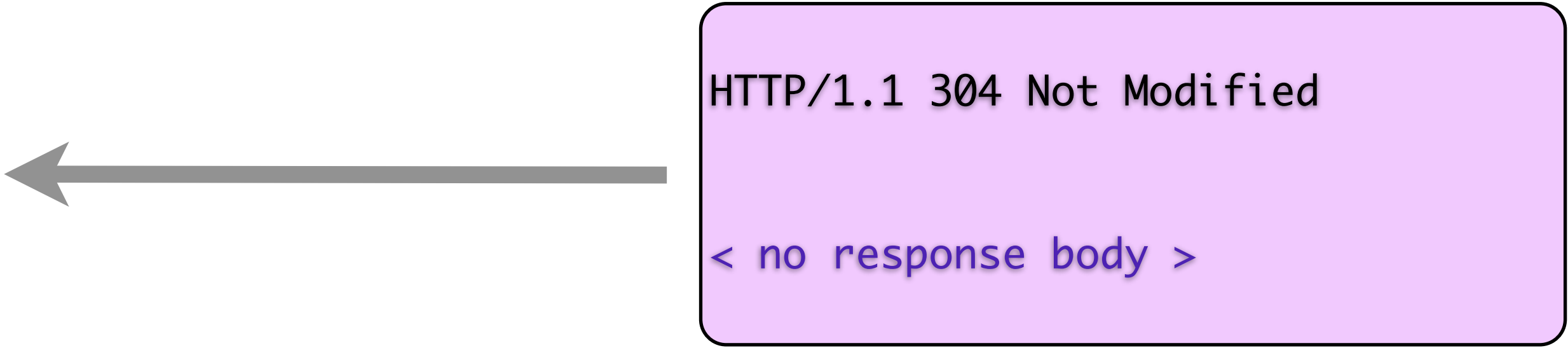
Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-Modified-Since: <date>
```



```
HTTP/1.1 304 Not Modified
```

```
< no response body >
```



Etag/Last Modified on the Server

Etag/Last Modified on the Server

```
def show
```

Etag/Last Modified on the Server

```
def show
  @band = Band.find(params[:id])
```

Etag/Last Modified on the Server

```
def show
  @band = Band.find(params[:id])
  fresh_when(:etag => @band,
```

Etag/Last Modified on the Server

```
def show
  @band = Band.find(params[:id])
  fresh_when(:etag => @band,
             :last_modified => @band,
```

Etag/Last Modified on the Server

```
def show
  @band = Band.find(params[:id])
  fresh_when(:etag => @band,
             :last_modified => @band,
             :public => true)
```

Etag/Last Modified on the Server

```
def show
  @band = Band.find(params[:id])
  fresh_when(:etag => @band,
             :last_modified => @band,
             :public => true)
  expires_in 10.minutes, :public => true
```

Etag/Last Modified on the Server

```
def show
  @band = Band.find(params[:id])
  fresh_when(:etag => @band,
             :last_modified => @band,
             :public => true)
  expires_in 10.minutes, :public => true
end
```

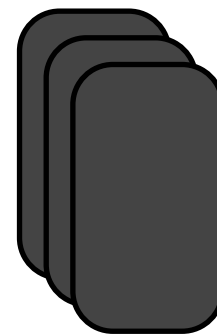
Etag/Last Modified on the Server

```
def show
  @band = Band.find(params[:id])
  fresh_when(:etag => @band,
             :last_modified => @band,
             :public => true)
  expires_in 10.minutes, :public => true
end
```

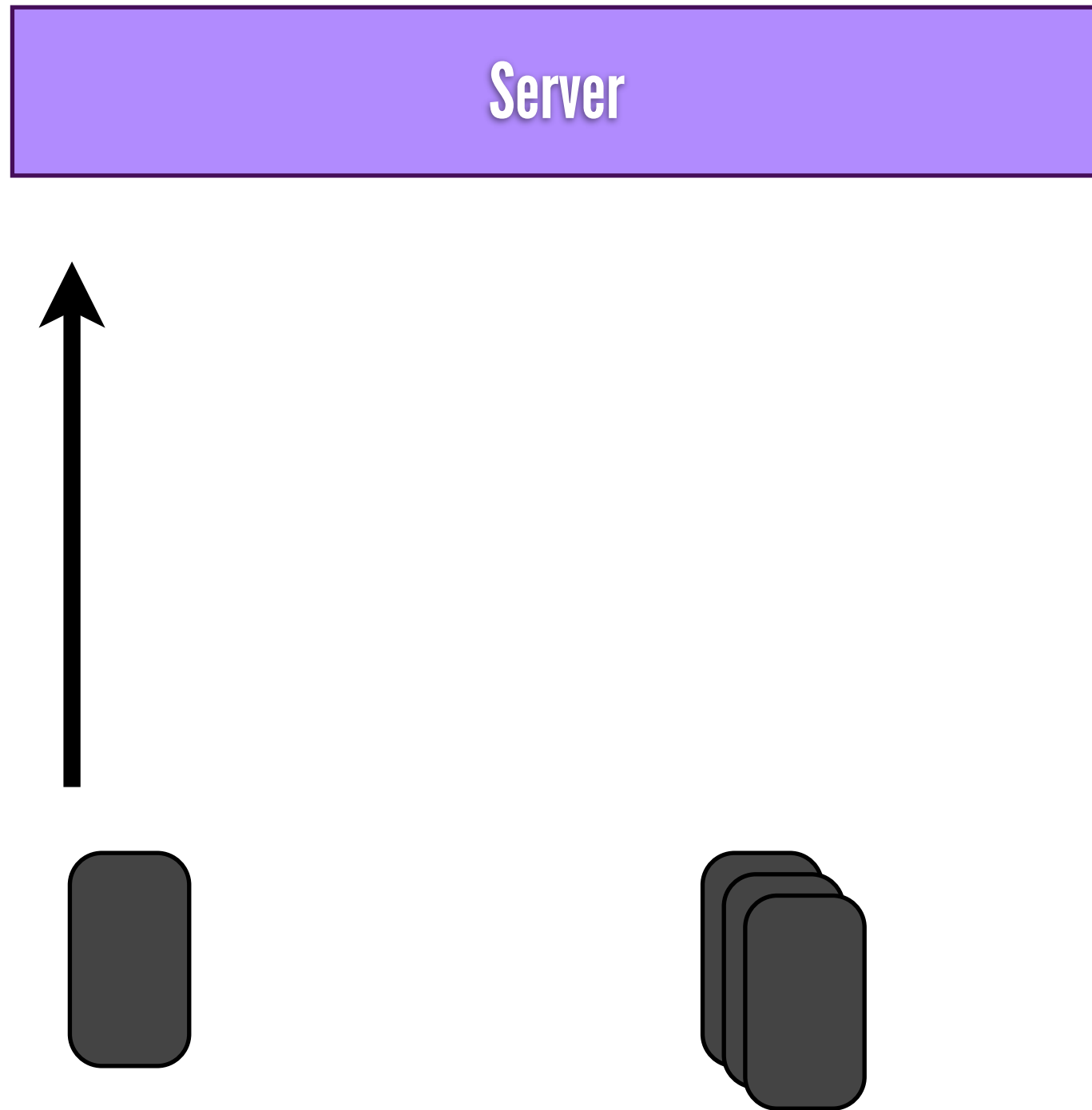
**Note: server still does a db query,
but doesn't render a template**

Saves on rendering & bandwidth for response transfer

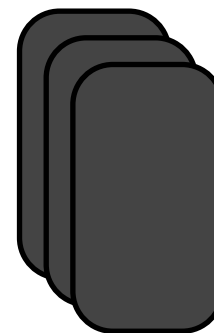
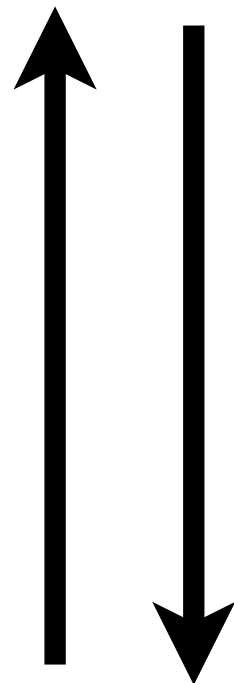
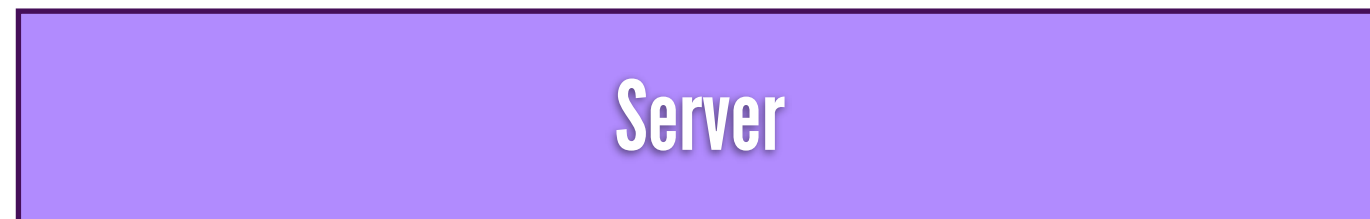
Cache-Control limitations



Cache-Control limitations

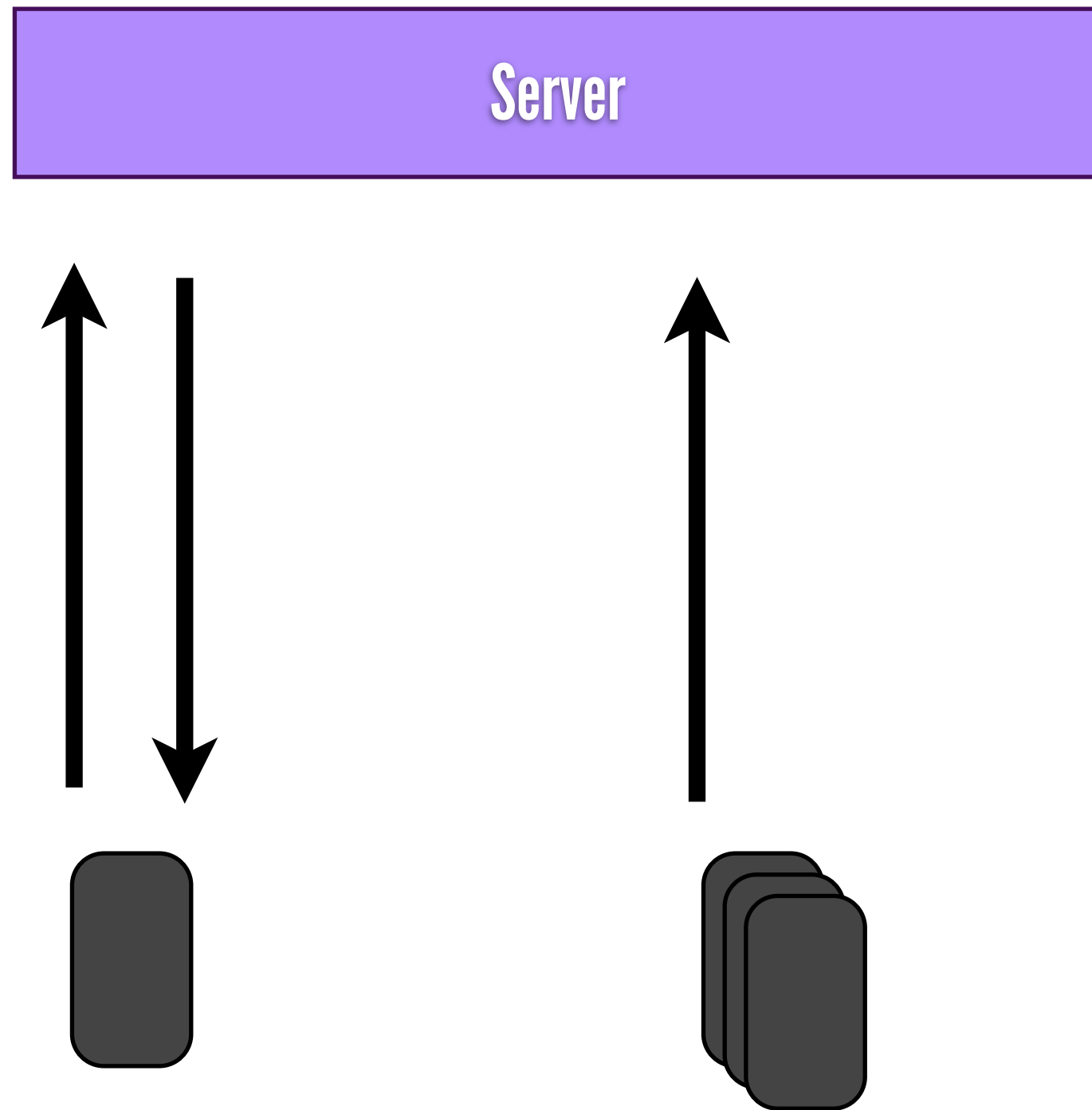


Cache-Control limitations



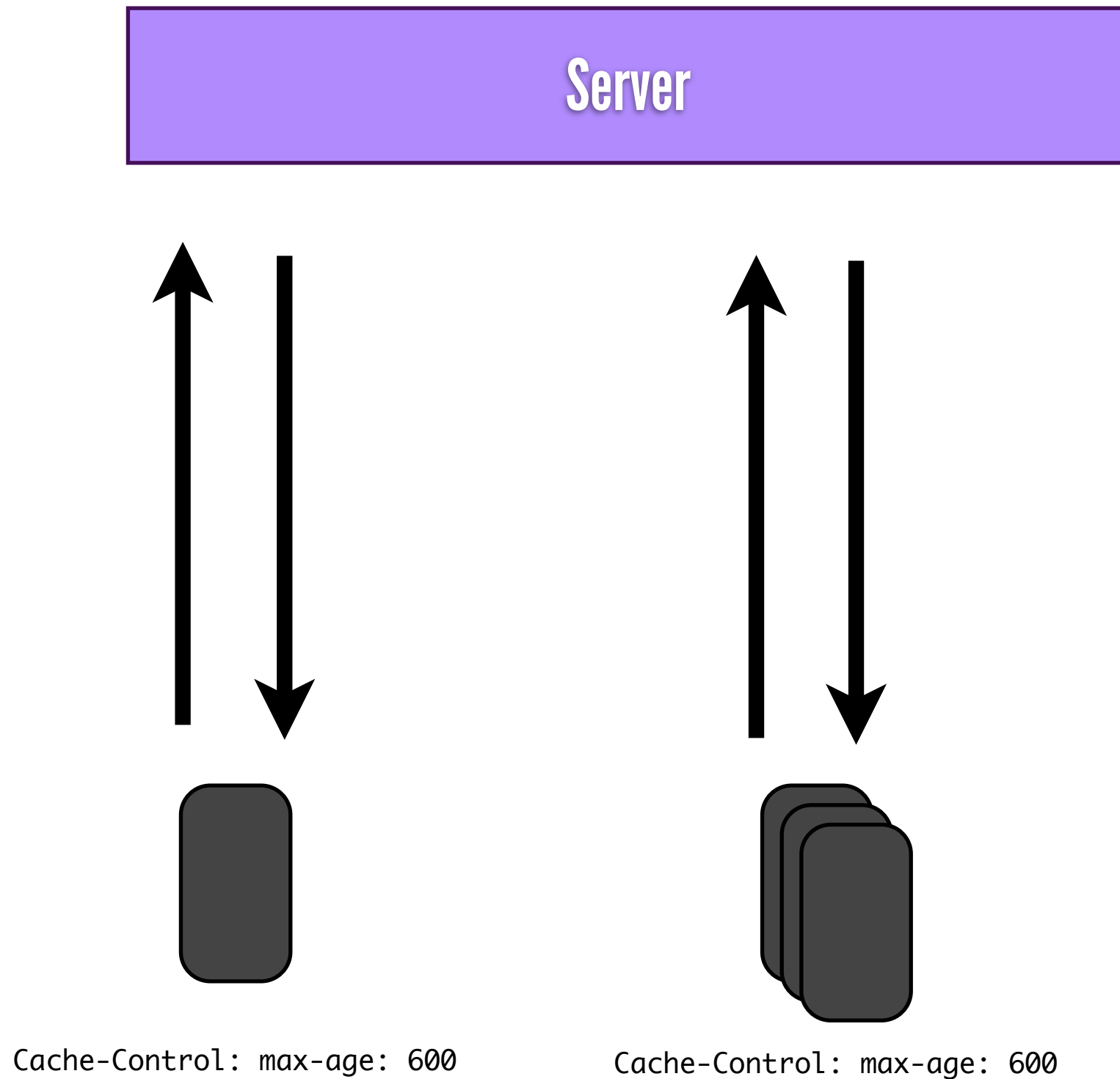
Cache-Control: max-age: 600

Cache-Control limitations



Cache-Control: max-age: 600

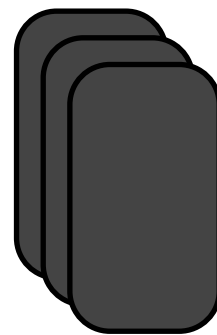
Cache-Control limitations



Reverse-Proxy Cache FTW

Server

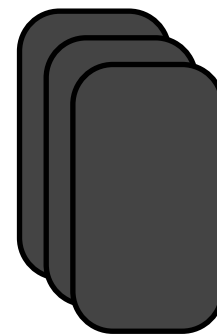
Varnish / Squid / Nginx



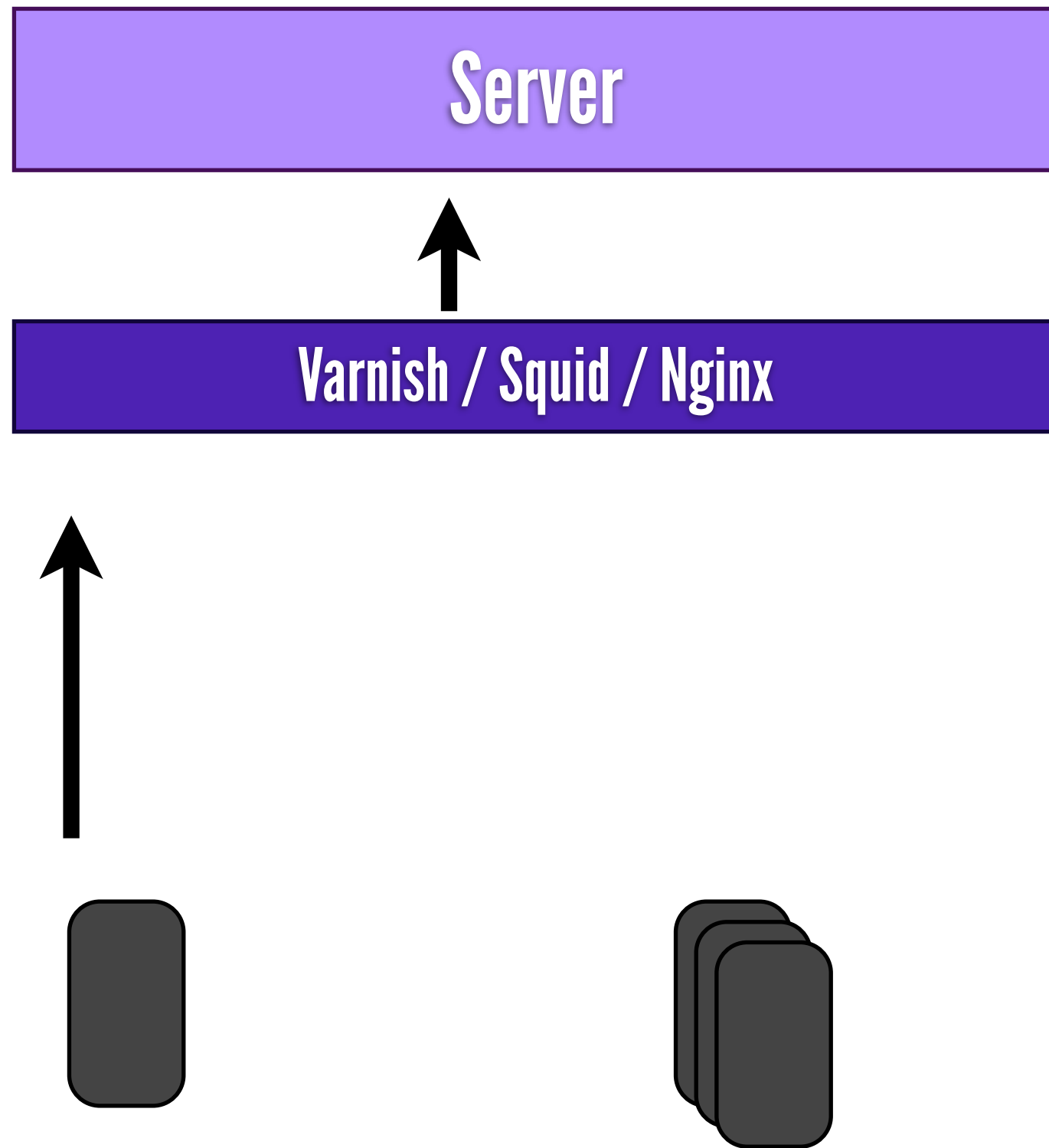
Reverse-Proxy Cache FTW

Server

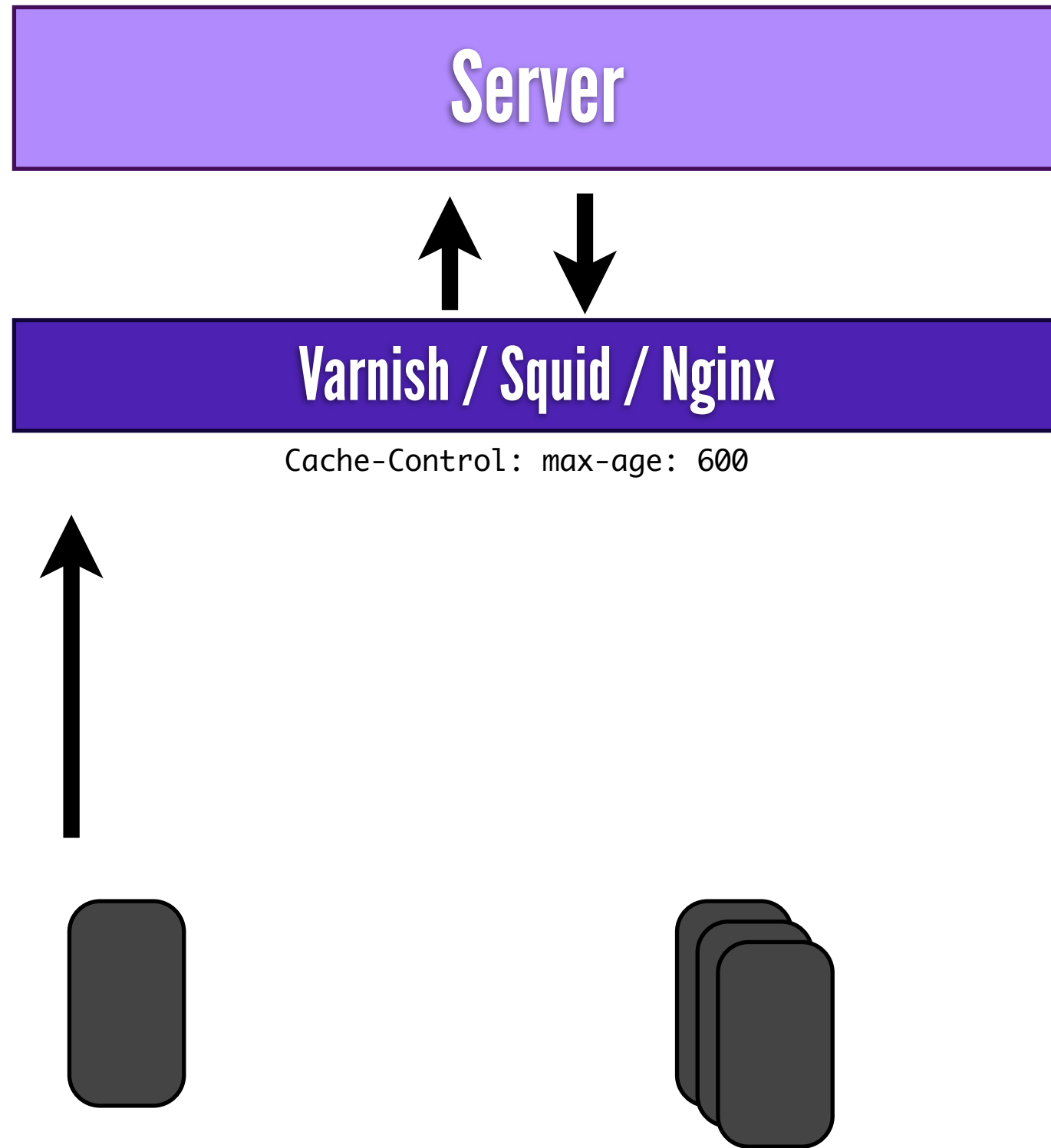
Varnish / Squid / Nginx



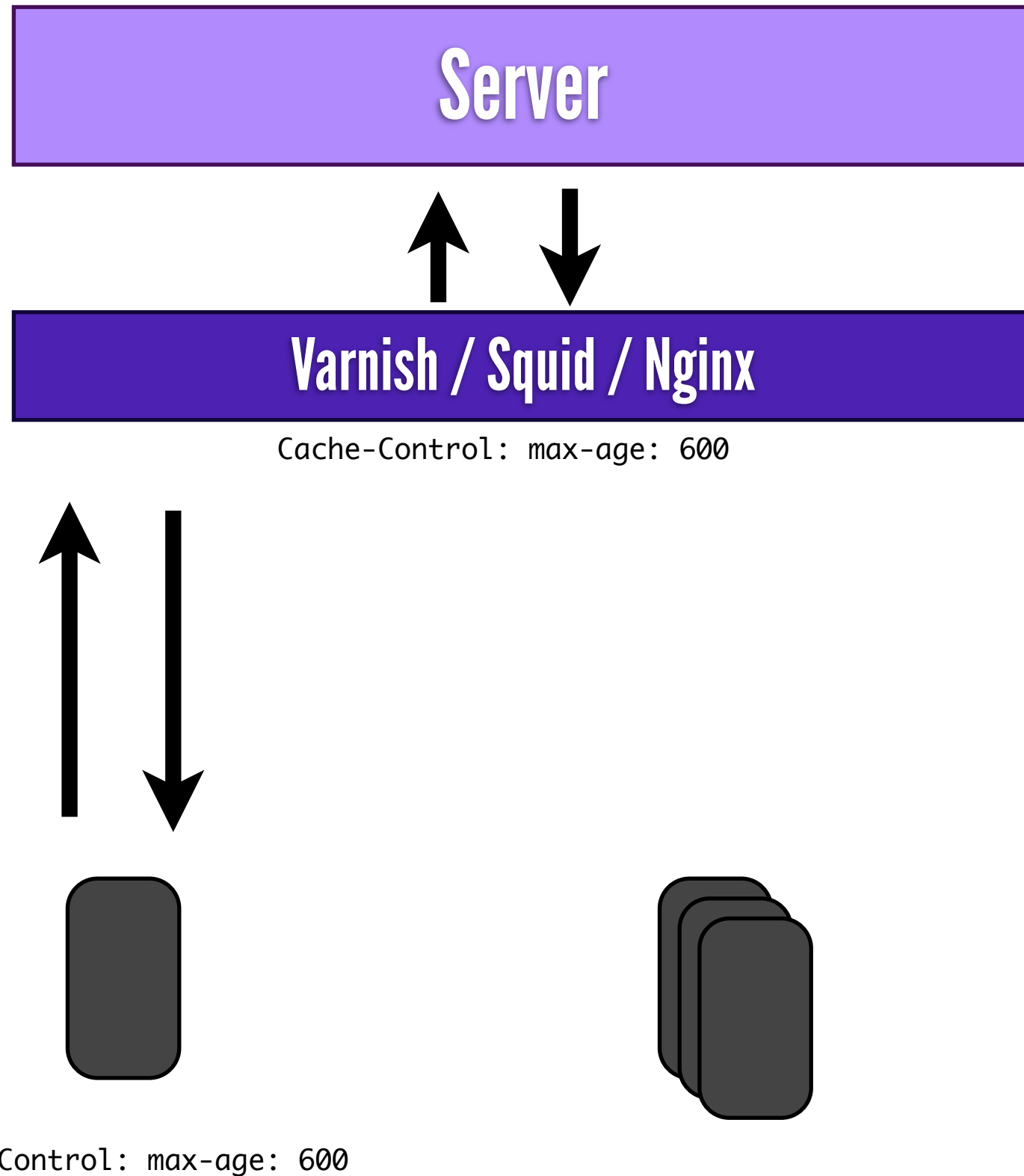
Reverse-Proxy Cache FTW



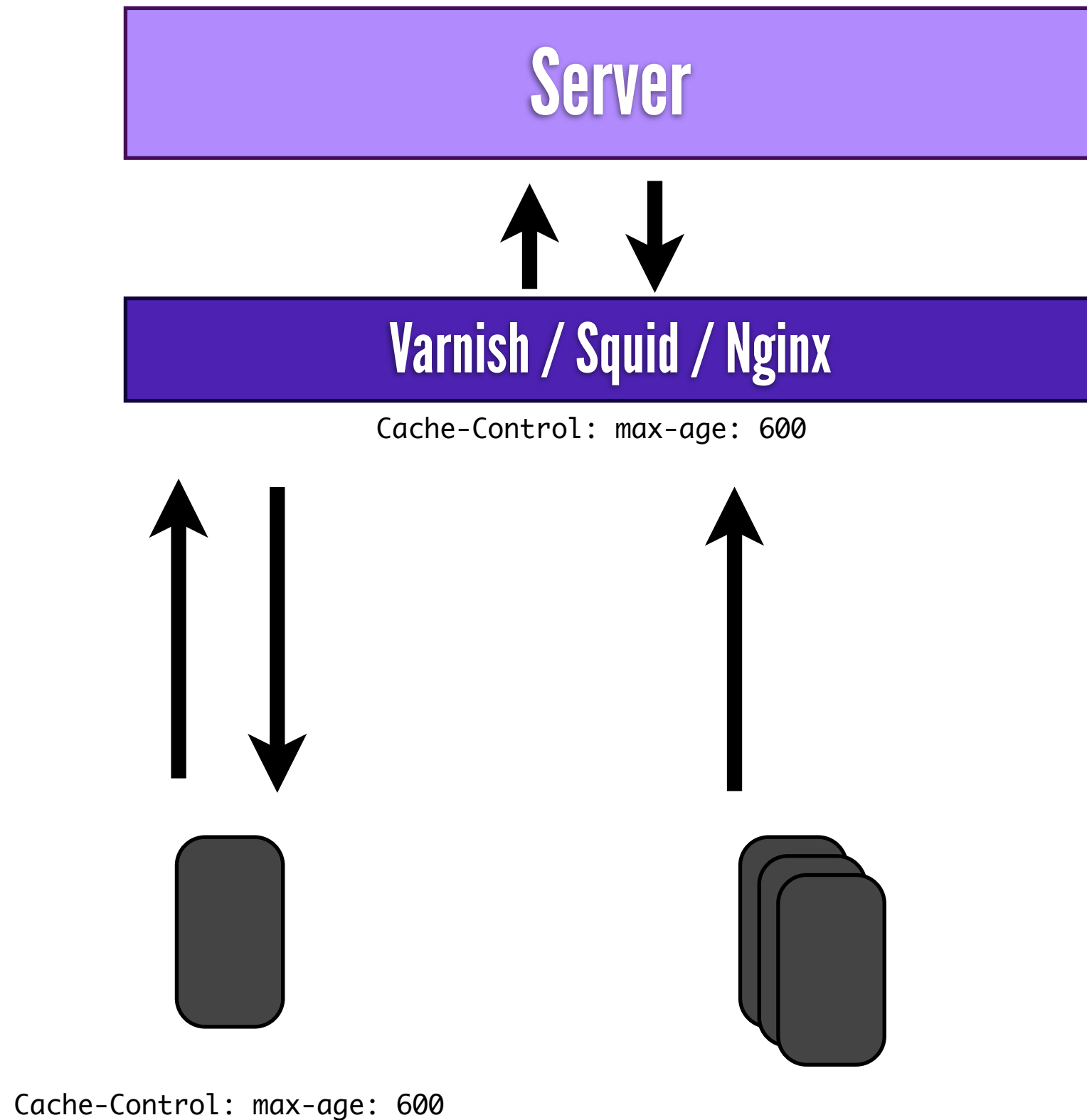
Reverse-Proxy Cache FTW



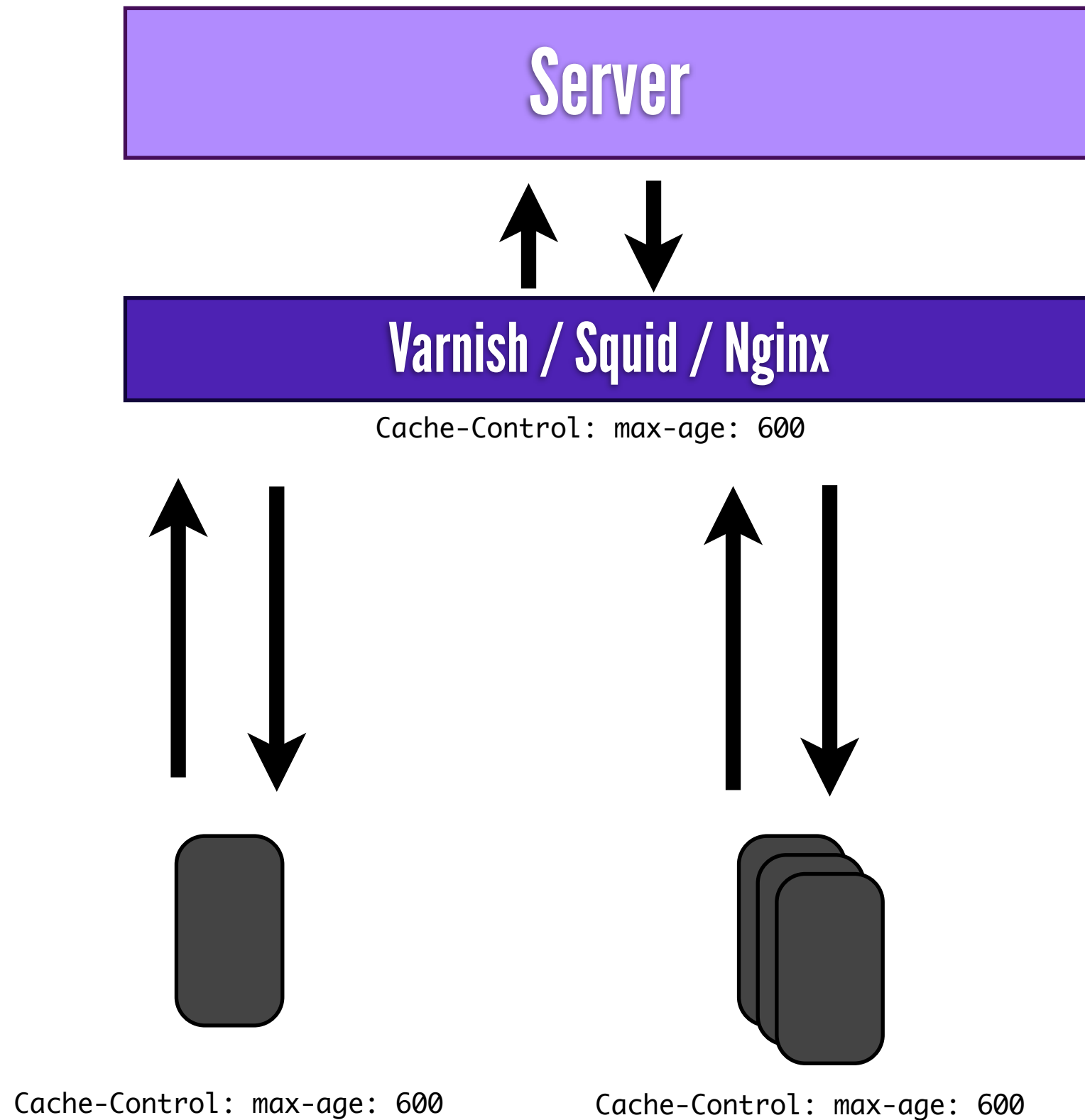
Reverse-Proxy Cache FTW



Reverse-Proxy Cache FTW



Reverse-Proxy Cache FTW



Leverage Cache-Control?



```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=9999999999, public
Date: Sun, 21 Oct 2012 16:15:37 GMT
```





Mike? Don't ever call me again.

Pros / Cons of Cache-Control

Client doesn't wait for response

Server spends no resources

But, clients may render stale data

Pros / Cons of Etag / IMS

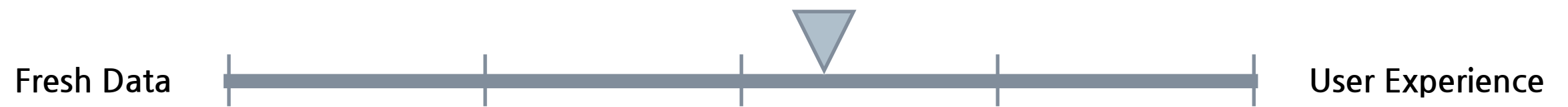
Server skips rendering
Miniscule data transfer
Appears instantaneous*

But, server still doing a db query

Reason about your data

- **List of US States**
- **User's Profile**
- **Customer's Order**
- **Activity Timeline**
- **Yesterday's stats**

Tradeoff



Caching w/ NSURLSession

**Leverages NSURLCache
already**

**Use default session configuration or customize cache
settings**

Use ephemeral configuration to disable caching

Caching w/ NSURLSession

Customize if needed

```
NSURLCache *cache =  
    [[NSURLCache alloc] initWithMemoryCapacity:1024*1024*10  
                                     diskCapacity:1024*1024*50  
                                     diskPath:nil];  
[sessionConfiguration setURLCache:cache];
```

Default Cache Location

~/Library/Caches/com.acme.app/Cache.db

Navicat Essentials for SQLite – trial version30 days left. [Buy Now!](#)

ConnectionTableViewIndexTriggerQuery

Cache

main

Tables

cfurl_cache_blob_data

cfurl_cache_receiver_data

cfurl_cache_response

cfurl_cache_schema_version

sqlite_sequence

Views

Indexes

Triggers

Queries

cfurl_cache_blob_data

cfurl_cache_receiver_data

cfurl_cache_response

cfurl_cache_schema_version

sqlite_sequence

cfurl_cache_response @main (Cache)

ImageTextHexImport WizardExport WizardSort AscendingSort DescendingRemove Sorting

	entry_ID	version	hash_value	storage_policy	request_key	time_stamp
	1	0	1424005679	0	https://alpha-api.	2012-10-19 17:2
	2	0	-1305208724	0	https://d2rfichhc	2012-10-19 17:3
	3	0	1572491858	0	https://d2rfichhc	2012-10-19 17:3
	4	0	-1997901260	0	https://d2rfichhc	2012-10-19 17:3
	5	0	-777154780	0	https://d2rfichhc	2012-10-19 17:3
	6	0	66153744	0	https://d2rfichhc	2012-10-19 17:3
	7	0	-404209947	0	https://d2rfichhc	2012-10-19 17:3
	8	0	-1931015091	0	https://d2rfichhc	2012-10-19 17:3
	9	0	338276037	0	https://d2rfichhc	2012-10-19 17:3
	10	0	1772610311	0	https://d2rfichhc	2012-10-19 17:3
	11	0	-769696739	0	https://d2rfichhc	2012-10-19 17:3
	12	0	-1381255225	0	https://d2rfichhc	2012-10-19 17:3
	13	0	2140559844	0	https://d2rfichhc	2012-10-19 17:3
	14	0	-358207116	0	https://d2rfichhc	2012-10-19 17:3
	15	0	-1622839865	0	https://d2rfichhc	2012-10-19 17:3
	16	0	1750647411	0	https://d2rfichhc	2012-10-19 17:3
	17	0	352022890	0	https://d2rfichhc	2012-10-19 17:3
	18	0	-1578321287	0	https://d2rfichhc	2012-10-19 17:3
	19	0	1302241355	0	https://d2rfichhc	2012-10-19 17:3
	20	0	1535007898	0	https://d2rfichhc	2012-10-19 17:3

select *, rowid "NAVICAT_ROWID" from "main"."cfurl_cache_response"1

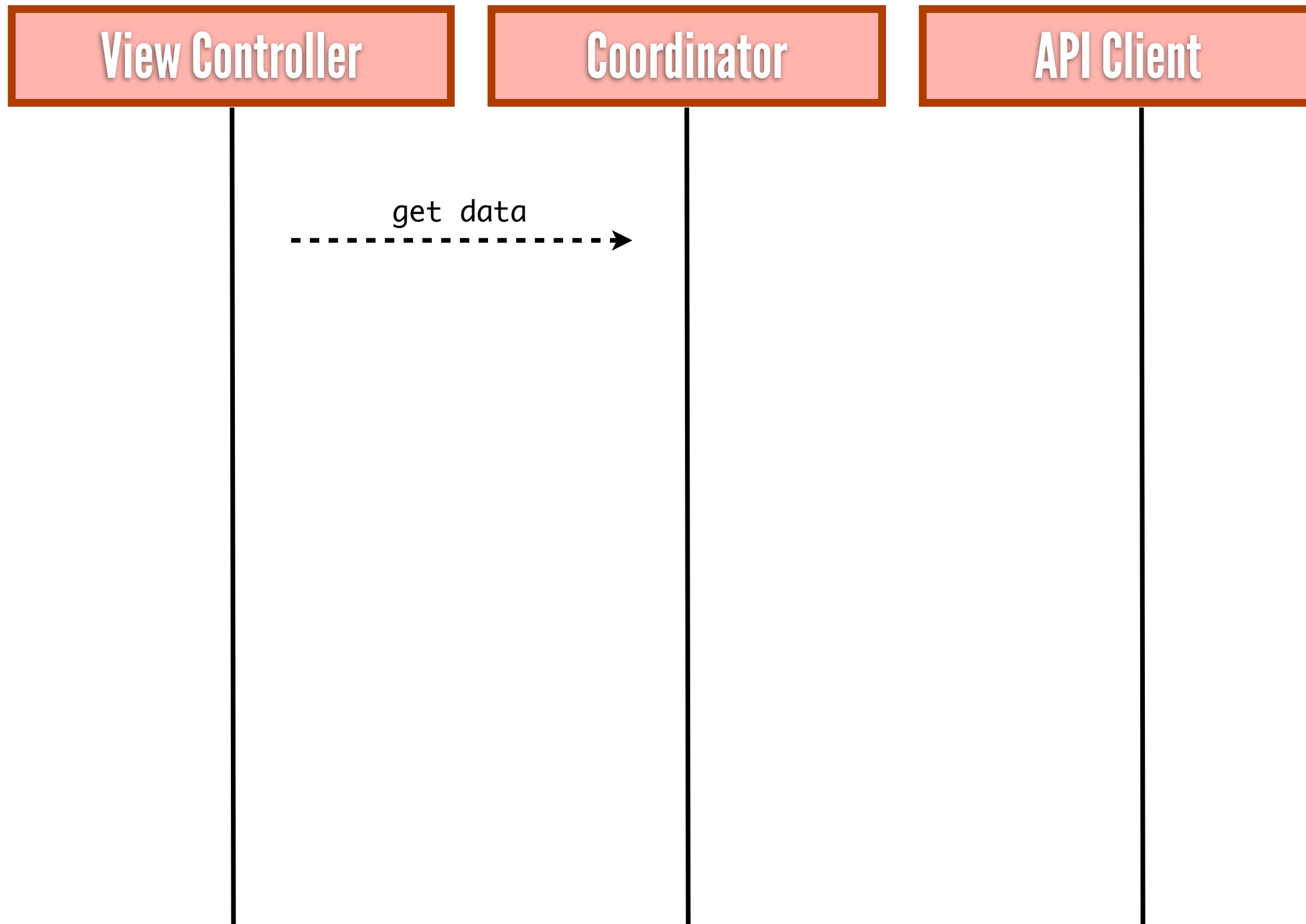
Switcheroo

View Controller

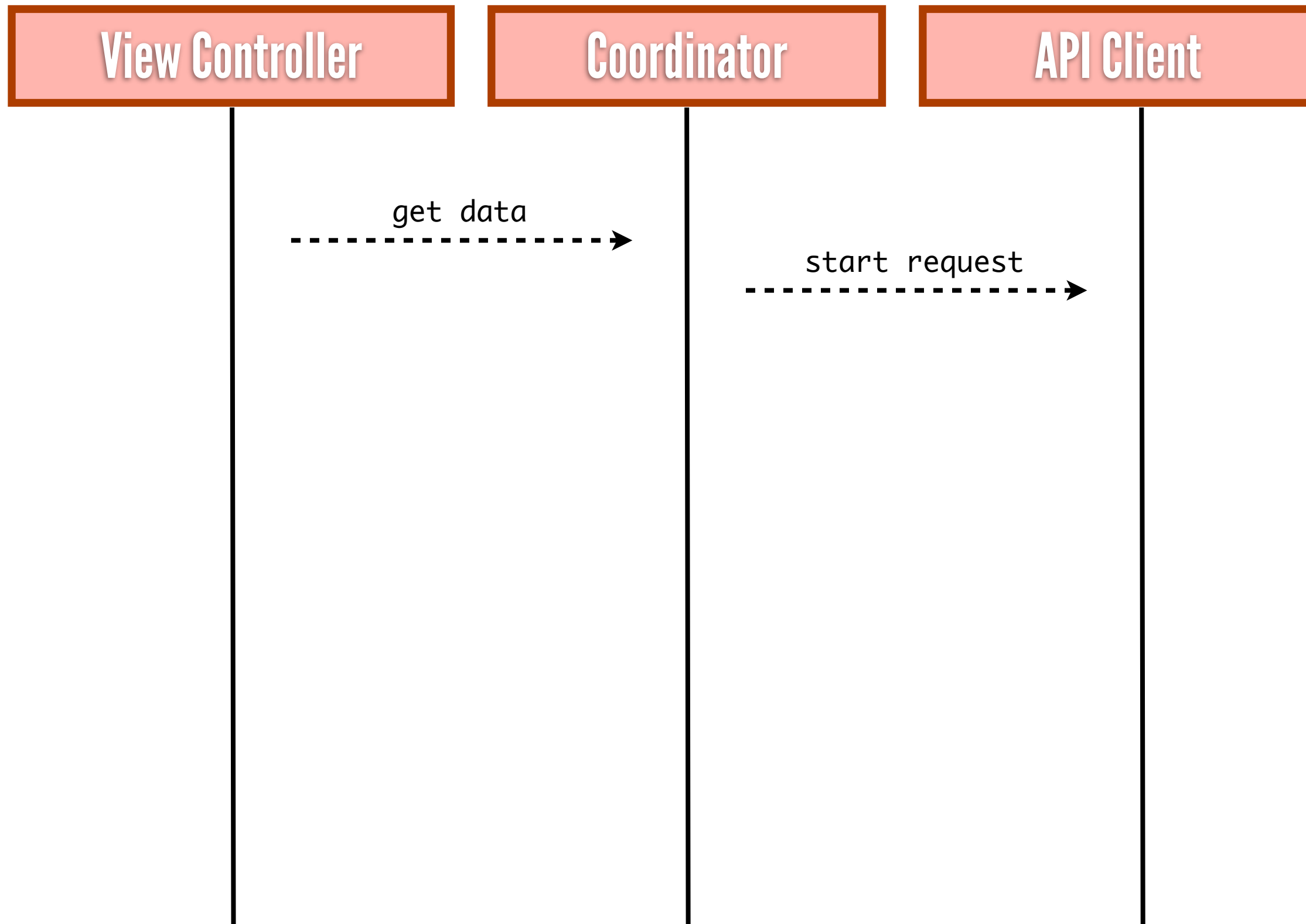
Coordinator

API Client

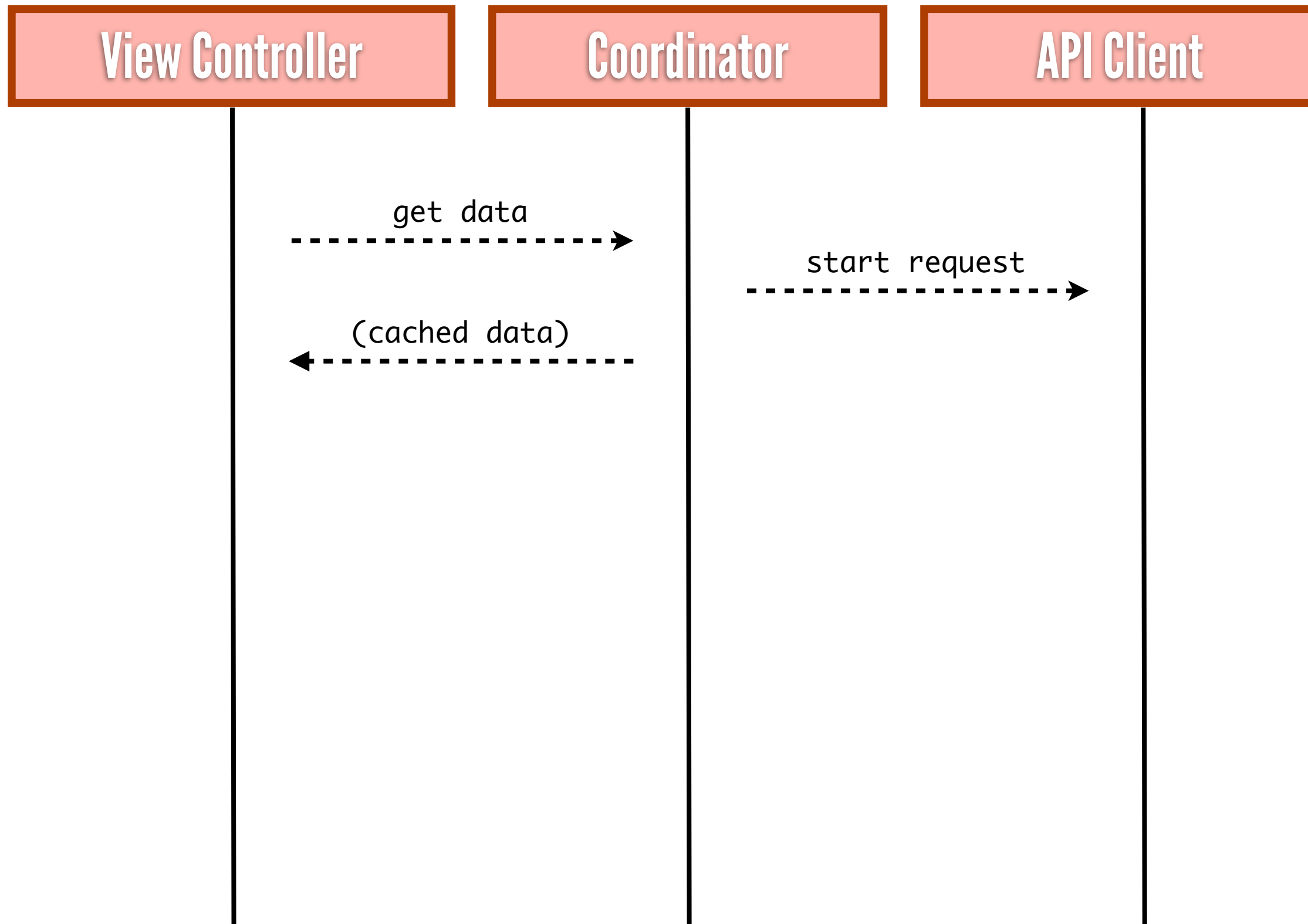
Switcheroo



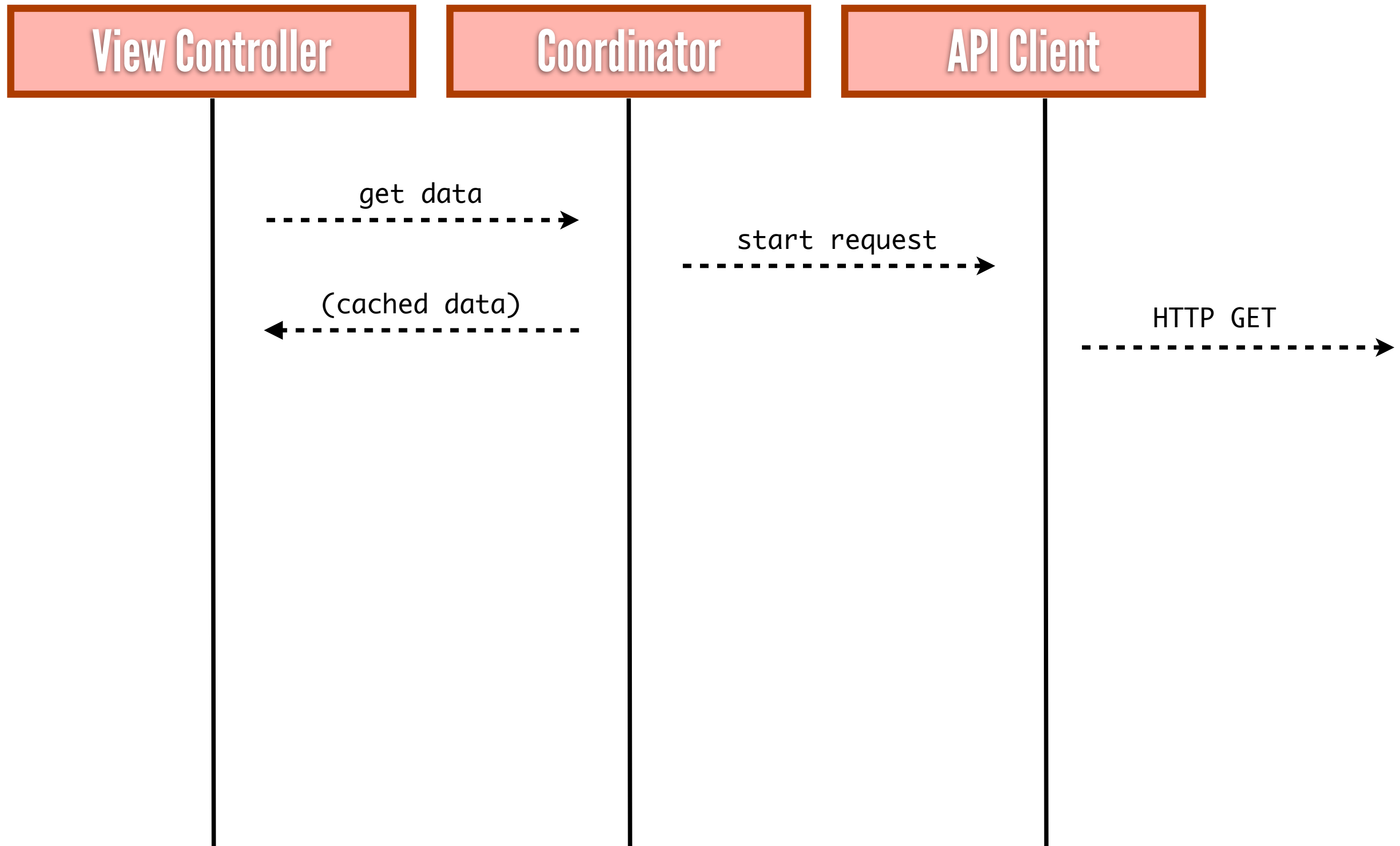
Switcheroo



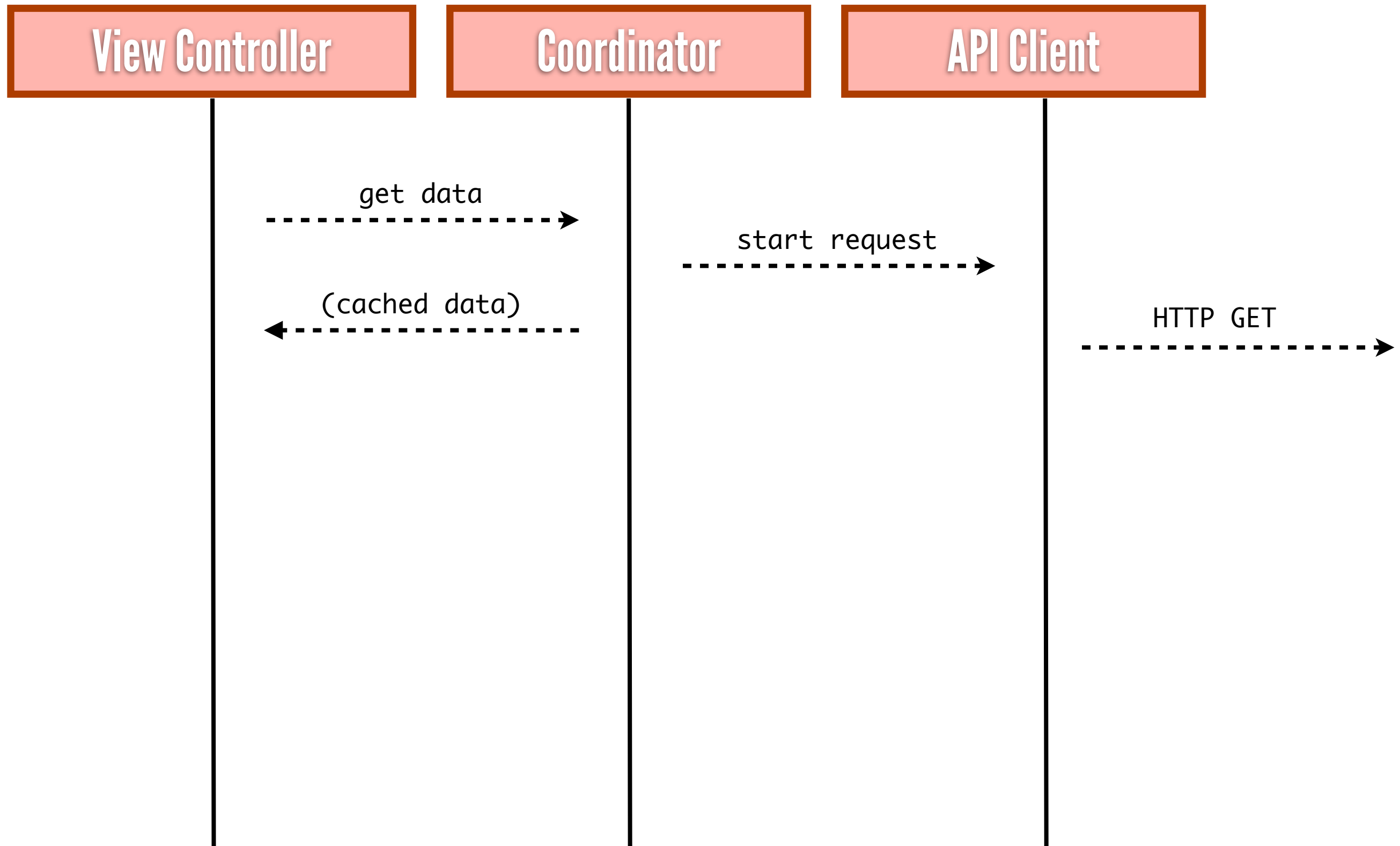
Switcheroo



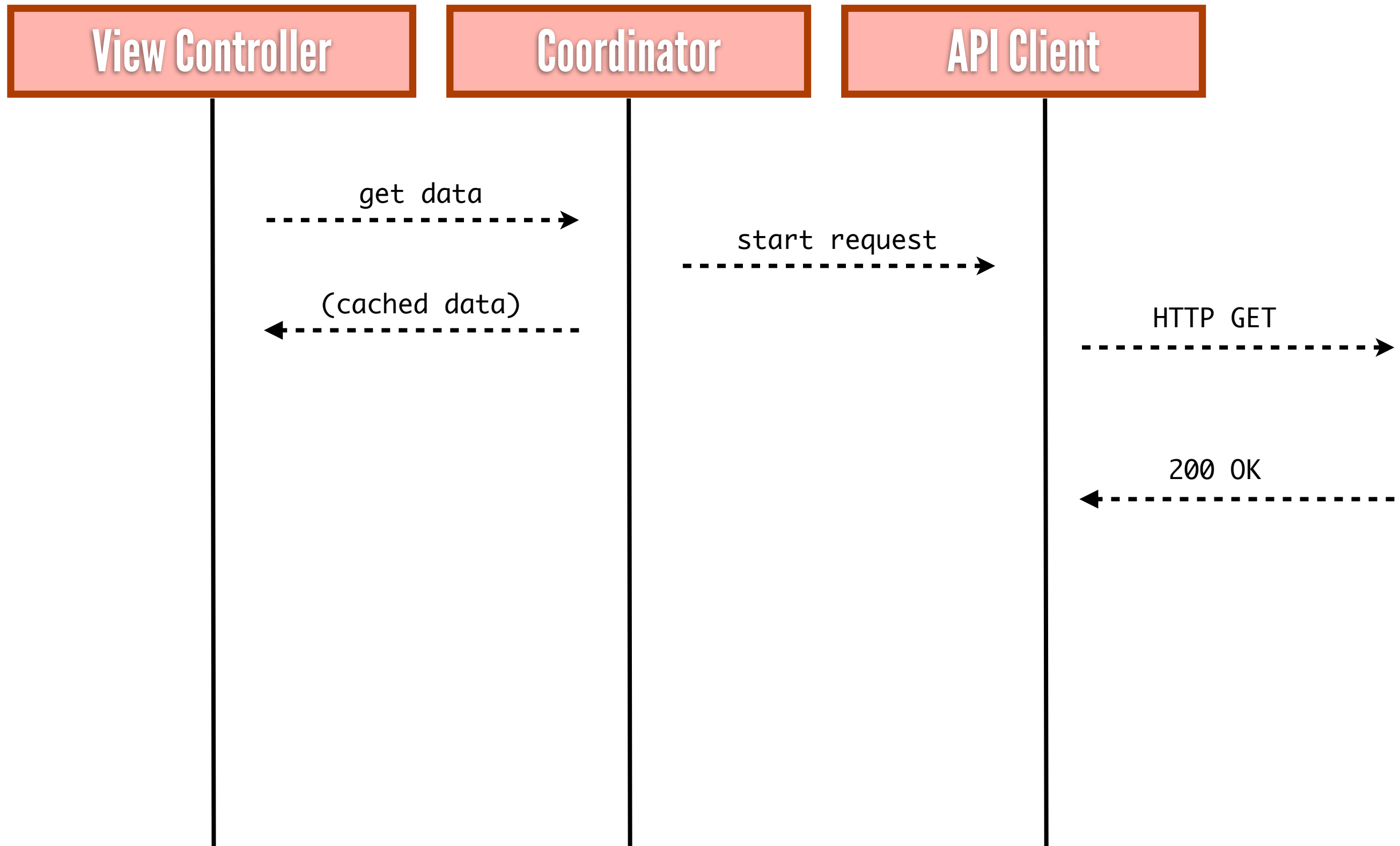
Switcheroo



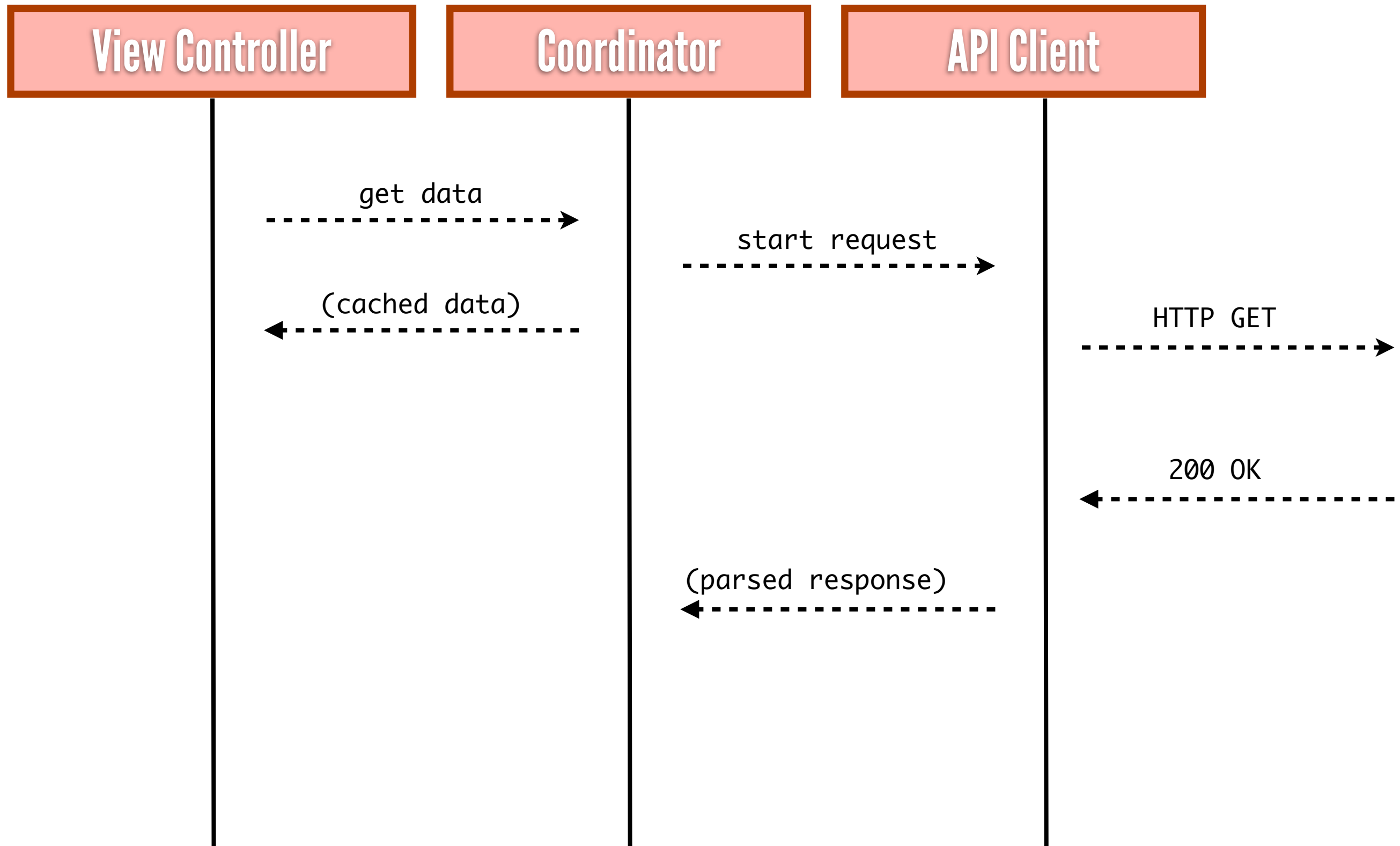
Switcheroo



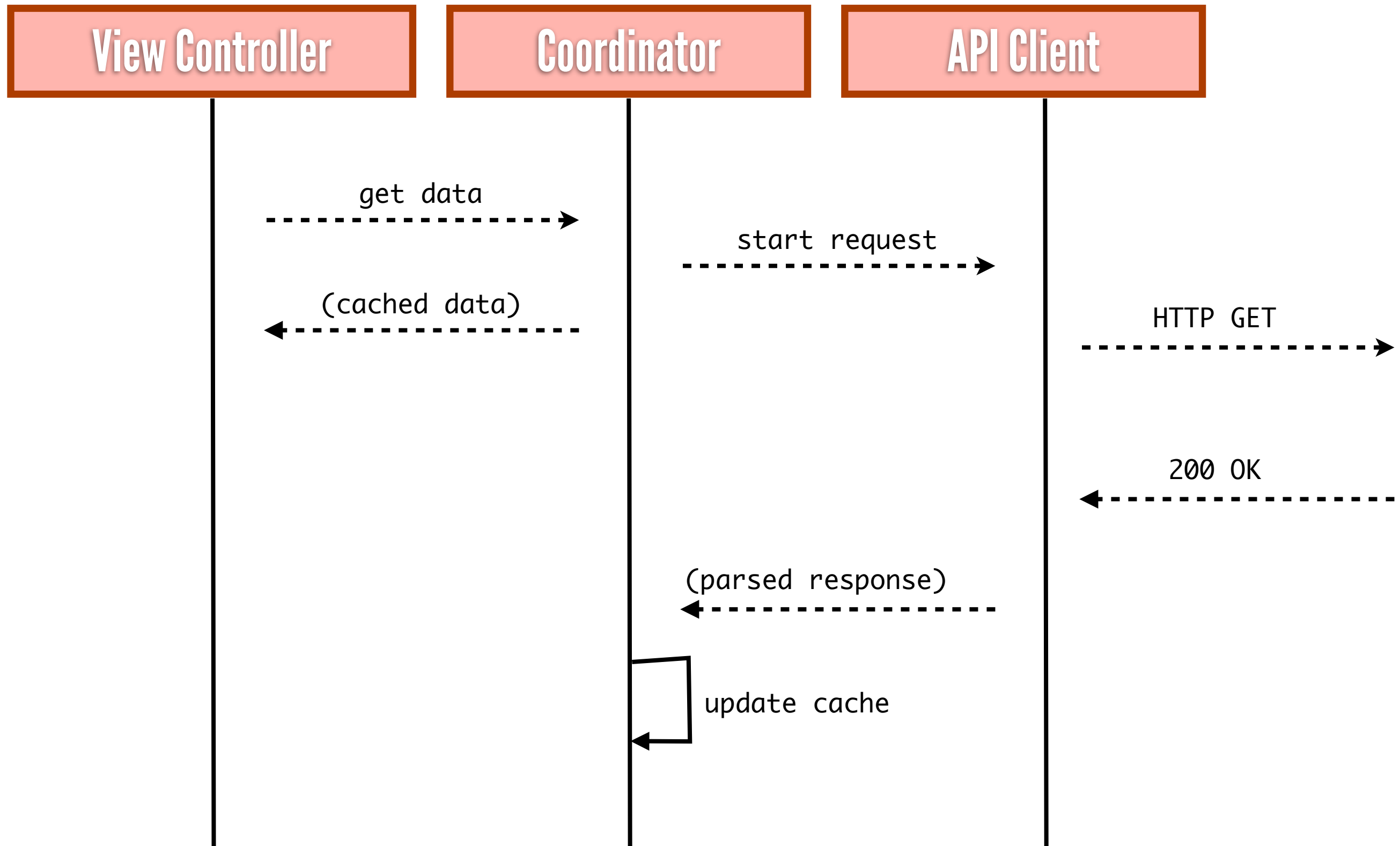
Switcheroo



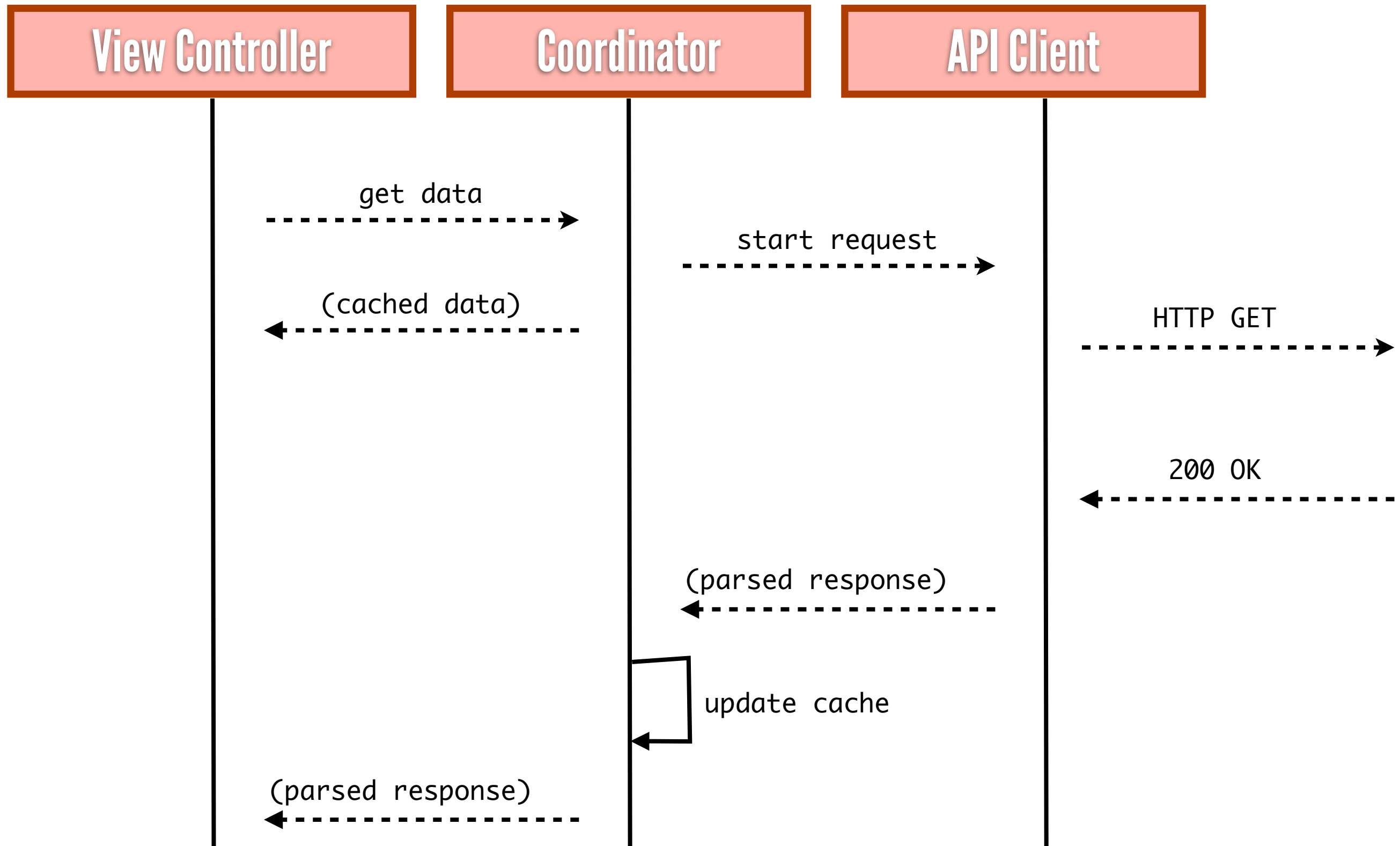
Switcheroo



Switcheroo



Switcheroo



API Design Tips

Don't expose your internal model



http://www.myspace.com/na6_build/photos/20613762

API Design Tips

Version your API



API Design Tips

**Send OS Version, App Version,
and Device Name as headers**



Device Name
OS Version
App Version

API Design Tips

**Have a way to notify clients
to upgrade (even force)**

API Design Tips

- **Techniques for permeating deletes**
 - **Valid ID Check**
 - **Client requests changes, server returns a set of new records, changed records, and deleted record ids**

API Design Tips

Page Large Datasets

API Design Tips

**Build-in HTTP Caching support
from day 1**

API Design Tips

**Take care when caching
user-specific data**

API Design Tips

Turn on Gzip Compression

API Design Tips

**Measure your API response
times**

Thank you!

Ben Scheirman

@subdigital

@bens (adn)

@nsscreencast