

An example of Future composition in a real app

Phil Calçado
SoundCloud



the

mothership



api.soundcloud.com



api.soundcloud.com

Search

Groups

Playlists

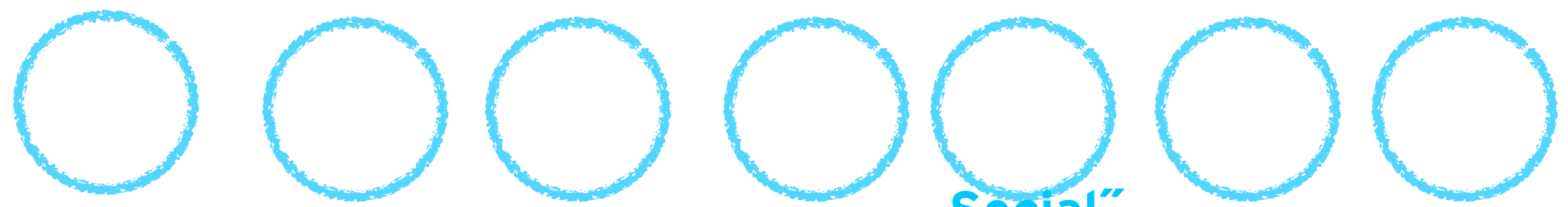
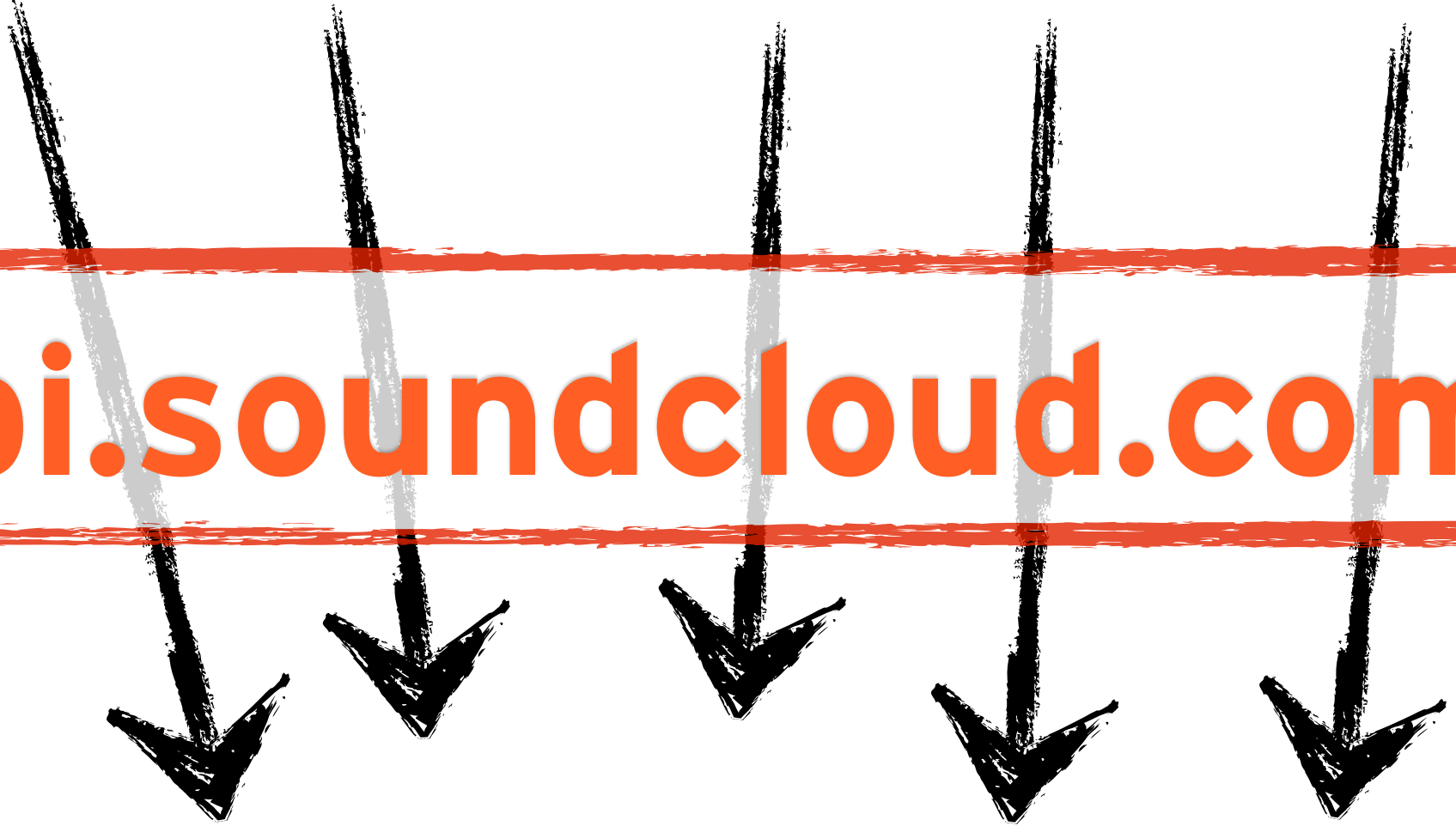
Msgs

Metadata

Stats



api.soundcloud.com



**Groups Metadata Playlists Search Social
Graph Messages Stats**

Rails

doesn't like

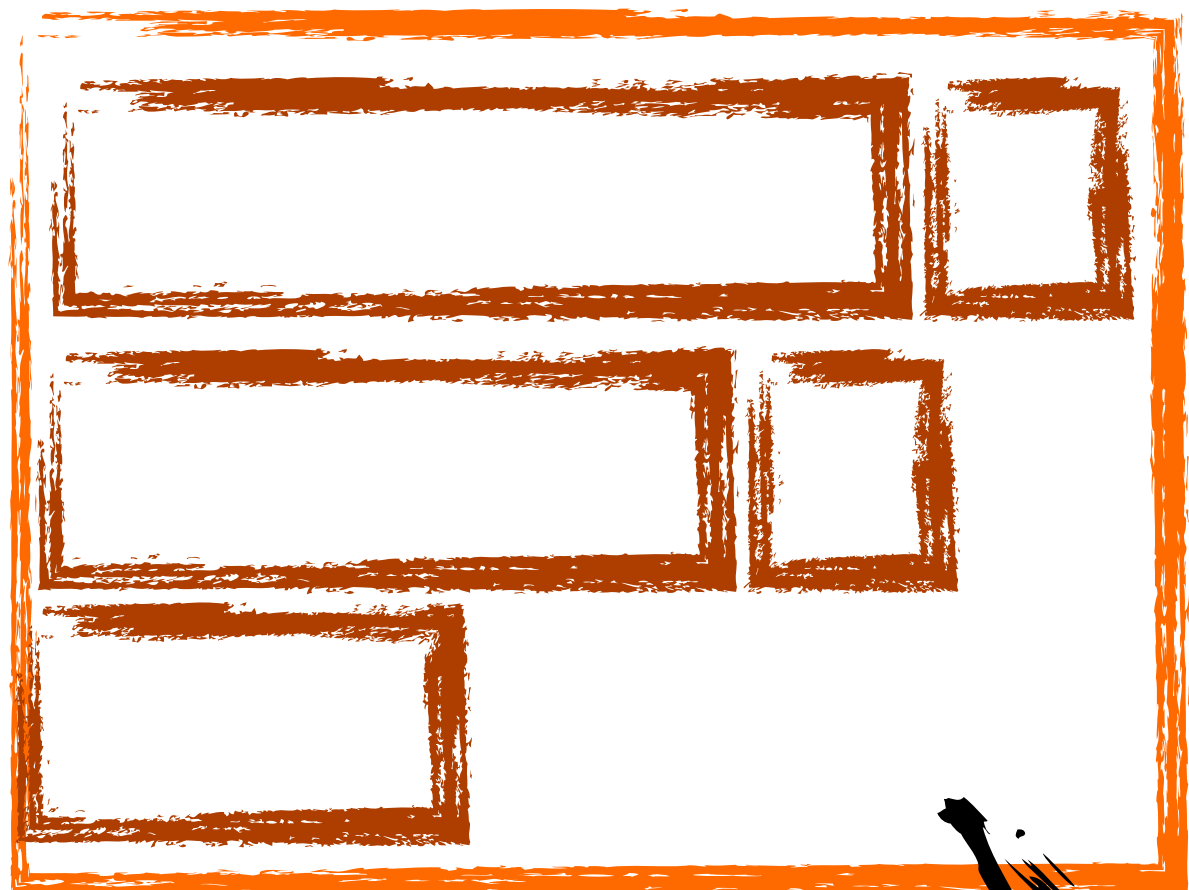
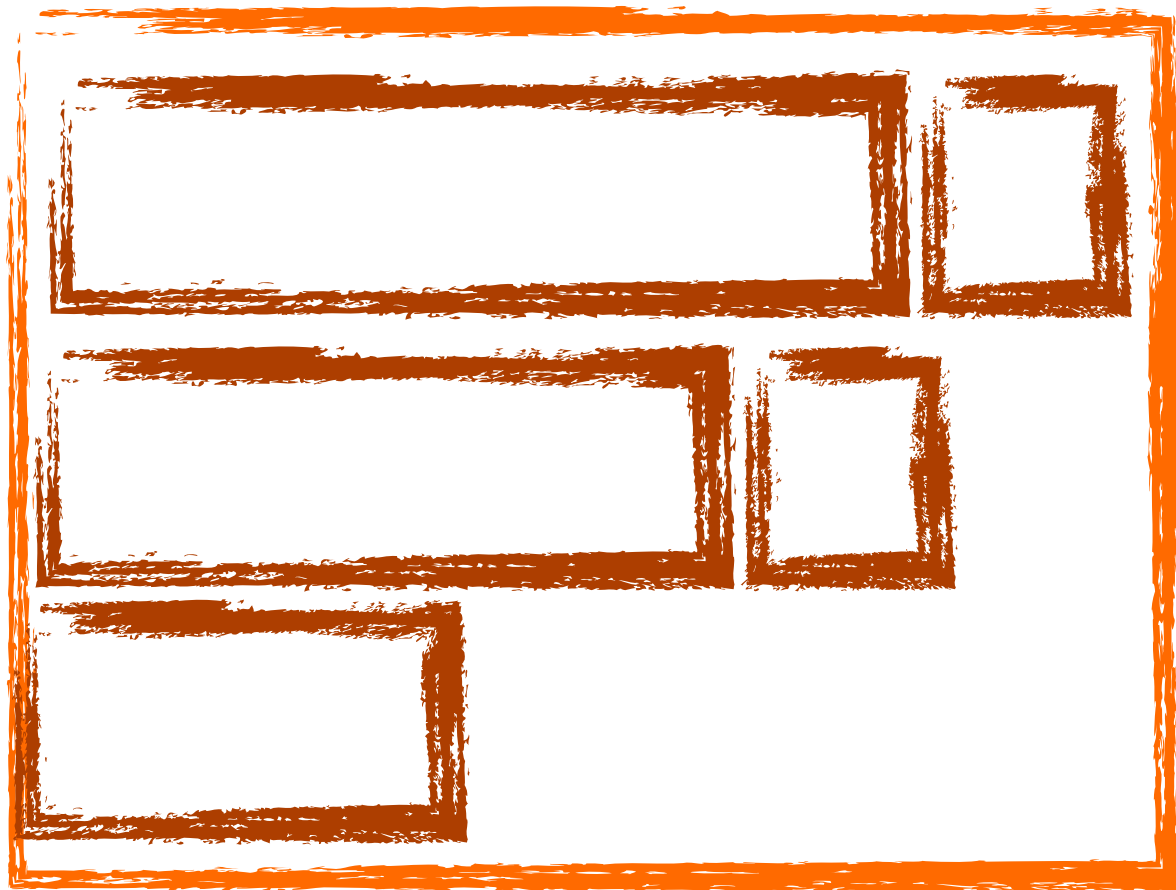
parallelism

parallelism is like



time

concurrency is like



time

ideal is like

time



rails is like



use case #1

sequential

```
def get_tracks_related_to(track_urn)
  related_tracks_urns = @recommender_client.get_related_to(track_urn)
  related_tracks_urns.take(5).map { |t| @metadata_client.get_data_for(t) }
end
```

use case #2

concurrent

```
def can_upload?(user_urn)
  warnings = @user_warnings_client.for_user(user_urn)
  spammer = @spam_detection_client.is_spammer?(user_urn)
  blocked = @blocked_users.is_blocked?(user_urn)
  warnings.empty? and (not spammer) and (not blocked)
end
```


enter



use case #1

sequential

```
def flatMap[B](f: A => Future[B]): Future[B]  
def map[B](f: A => B): Future[B] = flatMap { a => Future { f(a) } }
```

```
def byUrn(session: UserSession, urns: Seq[Urn]): Future[Seq[Track]] = {  
  urns.map { urn =>  
    fetchOnlyTracksByUrn(session, urn).flatMap { onlyAvailableTracksJson =>  
      buildFullTracks(session, onlyAvailableTracksJson)  
    }  
  }  
}
```

maybe more readable

```
def byUrn(session: UserSession, urns: Seq[Urn]): Future[Seq[Track]] = {  
  urns.map { urn =>  
    for {  
      onlyAvailableTracksJson <- fetchOnlyTracksByUrn(session, urn);  
      fullTracks <- buildFullTracks(session, onlyAvailableTracksJson)  
    } yield fullTracks  
  }  
}
```


use case #2

concurrent

```
/**
 * Take a sequence of Futures, wait till they all complete
 * successfully. The future fails immediately if any of the joined
 * Futures do, mimicking the semantics of exceptions.
 *
 * @param fs a java.util.List of Futures
 * @return a Future[Unit] whose value is populated when all of the fs return.
 */
def join[A](fs: java.util.List[Future[A]]): Future[Unit]

/**
 * Collect the results from the given futures into a new future of
 * Seq[A]. If one or of the given Futures is exceptional, the resulting
 * Future result will be the first exception encountered.
 *
 * @param fs a sequence of Futures
 * @return a Future[Seq[A]] containing the collected values from fs.
 */
def collect[A](fs: Seq[Future[A]]): Future[Seq[A]]
```



not concurrent!


```
def byUrn(session: UserSession, urns: Seq[Urn]): Future[Map[Urn, Playable]] = {  
  for {  
    allTracks <- tracksRepository.byUrn(session, urns.filter(_.getCollection == "tracks"));  
    allPlaylists <- playlistsRepository.byUrn(session, urns.filter(_.getCollection == "playlists"))  
  } yield (allTracks ++ allPlaylists).map(p => (p.urn, p)).toMap[Urn, Playable]  
}
```



ah!

```
def byUrn(session: UserSession, urns: Seq[Urn]): Future[Seq[Playable]] = {  
  val allTracks: Future[Seq[Track]] = tracksRepository.byUrn(session, urns.filter(_.getCollection == "tracks"))  
  val allPlaylists: Future[Seq[Playlist]] = playlistsRepository.byUrn(session, urns.filter(_.getCollection == "playlists"))  
  
  val allFutures = allTracks.join(allPlaylists)  
  
  allFutures.map {  
    case (tracks, playlists) => (tracks ++ playlists).map(p => (p.urn, p)).toMap[Urn, Playable]  
  }  
}
```


example


[Home](#)[Explore](#)[Sign in or](#)[Sign up](#)[Upload](#)

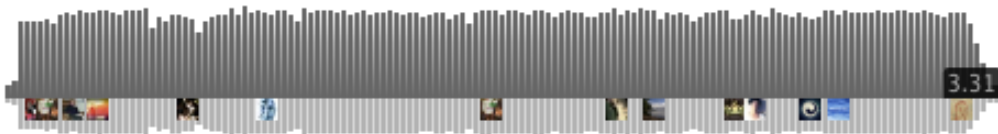


Phil Calçado 
Phil Calçado
Berlin, Germany
1,250 | 23
[Follow](#) 







Developer @ SoundCloud
[PhilCalcado.com](#)
[pcalcado](#)
[pcalcado](#)
[pcalcado](#)





Heimatt.  Phil Calçado
Everyone's a Sinner 1 day [#Folk](#)




3.31






    ▶ 2,438 | ♥ 81 |  28 |  14








Zap Records  Phil Calçado
Daniel Wilson - Young Rubbish EP 1 day [#Young Rubbish](#)




15.04


    iTunes ♥ 111 |  47


 Please Dream Again	▶ 6,889
 Trigger Dance	▶ 4,117
 Will You	▶ 10,355
 Young Rubbish	▶ 2,208
 If You Talk	▶ 1,656


[View all tracks >](#)




Sign up
Discover and connect
with the world's largest
community of music &
audio creators.




 35 playlists [View all](#)


 Phil Calçado
Stuff I found today
♥ 1

 Phil Calçado
Liza meets Janine

 Phil Calçado
Tarde de Sol
♥ 1

♥ 871 likes [View all](#)



example

example

50 or so 10-40ms requests

```
2014-07-07_12 INFO [PURCHASE] - [9 ms] GET /tracks/soundcloud:tracks:4121006/purchase_links -> 200 OK (soundcloud:applic
2014-07-07_12 INFO [PURCHASE] - [10 ms] GET /tracks/soundcloud:tracks:108647886/purchase_links -> 200 OK (soundcloud:app
2014-07-07_12 INFO [PURCHASE] - [12 ms] GET /tracks/soundcloud:tracks:10631430/purchase_links -> 200 OK (soundcloud:appl
2014-07-07_12 INFO [PURCHASE] - [12 ms] GET /tracks/soundcloud:tracks:28651320/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO [PURCHASE] - [9 ms] GET /tracks/soundcloud:tracks:17671873/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO [PURCHASE] - [11 ms] GET /tracks/soundcloud:tracks:93518877/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO [PURCHASE] - [11 ms] GET /tracks/soundcloud:tracks:5004134/purchase_links -> 200 OK (soundcloud:applic
2014-07-07_12 INFO [PURCHASE] - [8 ms] GET /tracks/soundcloud:tracks:112244513/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO [VISUALS] - [7 ms] GET /visuals?urn=soundcloud%3Atracks%3A23949338 -> 200 OK (soundcloud:applications
2014-07-07_12 INFO [PURCHASE] - [16 ms] GET /tracks/soundcloud:tracks:6183904/purchase_links -> 200 OK (soundcloud:applic
2014-07-07_12 INFO [PURCHASE] - [16 ms] GET /tracks/soundcloud:tracks:3632399/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO [PURCHASE] - [13 ms] GET /tracks/soundcloud:tracks:10631916/purchase_links -> 200 OK (soundcloud:appl
2014-07-07_12 INFO [PURCHASE] - [19 ms] GET /tracks/soundcloud:tracks:20474067/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO [PURCHASE] - [14 ms] GET /tracks/soundcloud:tracks:122846830/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO [PURCHASE] - [20 ms] GET /tracks/soundcloud:tracks:5625837/purchase_links -> 404 Not Found (soundclou
2014-07-07_12 INFO API-V2 - [152 ms] GET /profile/soundcloud%3Ausers%3A200380?offset=6&limit=50 -> 200 OK (unknown)%
```

total response time is 152ms

interesting

challenge:

**how to detect/type missed
concurrency opportunities?”
aka. for comprehension abuse**

phil calçado

**<http://philcalcado.com>
@pccalcado**

www.soundcloud.com

