Anatomy of Ruby I18n

Optional Features & Library Design

Ruby 118n

Official Hymne: Painkiller (Judas Priest)

Planets devastated
Mankinds on its knees
A saviour comes from out the skies
In answer to their pleas

This is the painkiller

With mankind ressurrected
Forever to survive
Returns from armageddon
to the skies

Anatomy of Ruby I18n

Optional Features & Library Design

About me

Sven F*cks

according to RailsEnvy ;-)

Sven Fuchs

```
$ say "phooks"
```

Sven Fuchs

- Berlin, employed at adva-business
- Learned Assembler on C64 in ~ 1984
- (Web) Developer since ~ 1996
- Rails I18n project since ~ 3 years

Sven Fuchs

http://svenfuchs.com

http://github.com/svenfuchs

http://twitter.com/svenfuchs

I18n 0.4.0

I18n 0.4.0

- Generic key/value backend

  Redis
  Tokyo Cabinet
  or ... any other key/value storage

I18n 0.4.0

- Generic key/value backend
- Transliterations

```
I18n.transliterate("Ümlaut!")
# => Uemlaut!
```

I18n 0.4.0

- Generic key/value backend
- Transliterations
- Deprecate {{foo}} in favor of %{foo}

Ruby 1.9 style interpolations

I18n 0.4.0

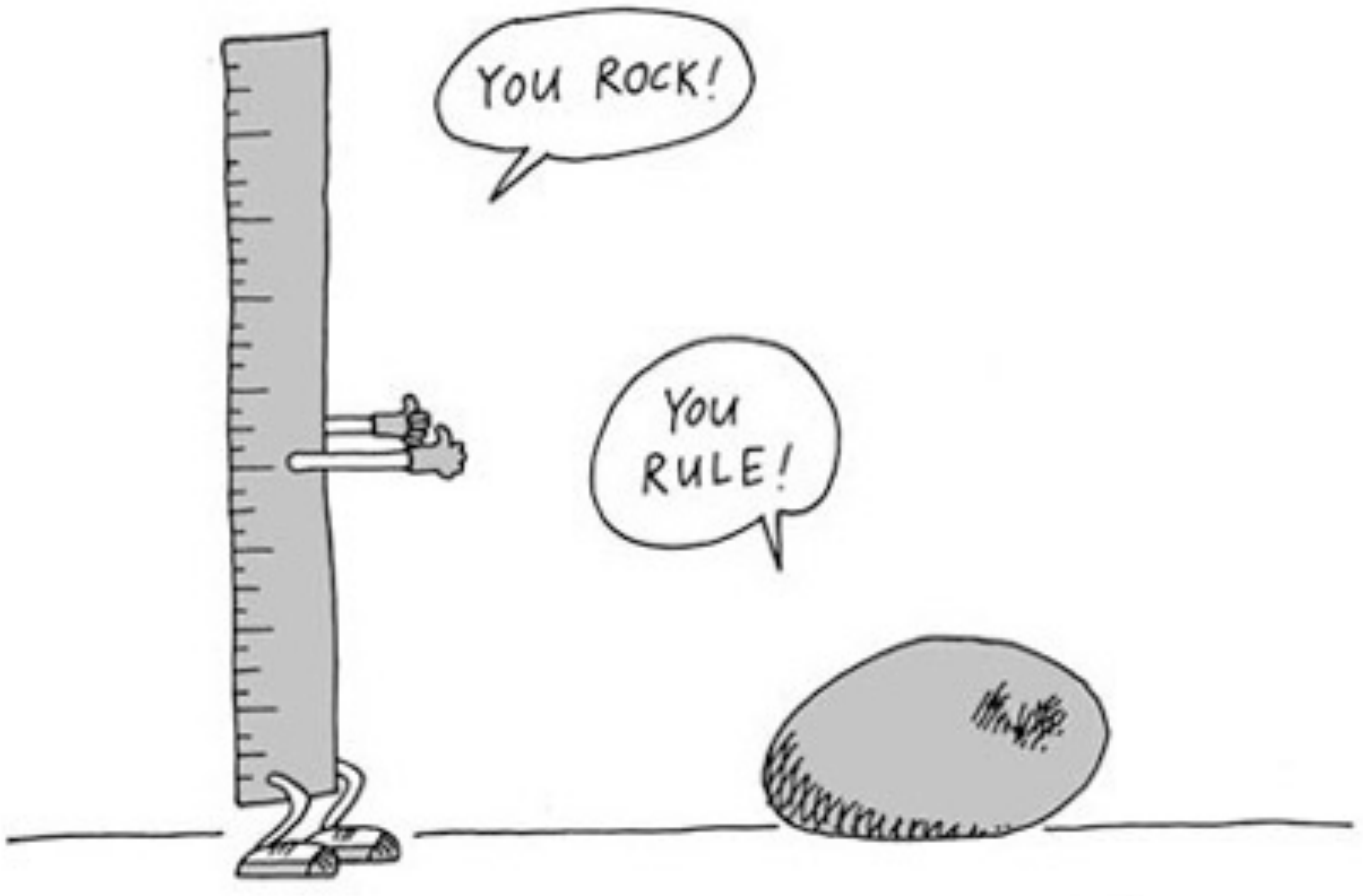
- Generic key/value backend
- Transliterations
- Deprecate {{foo}} in favor of %{foo}
- Refactorings
- Speed!

I18n 0.4.0

%{release_name}

I18n 0.4.0

Jose Valim rocks,
Norman Clarke rules


YOU ROCK!
YOU RULE!

Anatomy of Ruby I18n

Optional Features & Library Design

Anatomy of Ruby I18n

- What's I18n about?
- Optional modules and features
- Library design
- Patterns used

Questions

At the end of the talk, please :)

Overview

What’s this all about?

Definition

Internationalization is the process of designing software (...) so that it can be adapted to various languages and regions (...)

Localization is the process of adapting inter-nationalized software for a specific region or language (...)

http://en.wikipedia.org/wiki/Internationalization_and_localization

Put differently ...

```ruby
class Internationalization < Abstraction
  def perform
    @developer.work!
  end
end

class Localization < Concretion
  def perform
    @translator.work!
  end
end
```

Scope

Problems to solve

Scope

- Looking up translations
- Formats: numbers, dates, times, currency ...
- Timezones, calendar systems
- Collation (sorting)
- Character encodings

Scope

- ✓ Looking up translations
- ✓ Formats: numbers, dates, times, currency ...
- Timezones, calendar systems
- Collation (sorting)
- Character encodings

Scope

✓ Looking up translations
  - Interpolation
  - Pluralization
  - Defaults
  - Namespaces

✓ Formats: numbers, dates, times, currency ...

Scenarios

Contexts of I18n

Scenarios

- Single language
- Multiple languages
- Different storage types
- Model/Data translations
- Lot’s of special requirements

Requirements

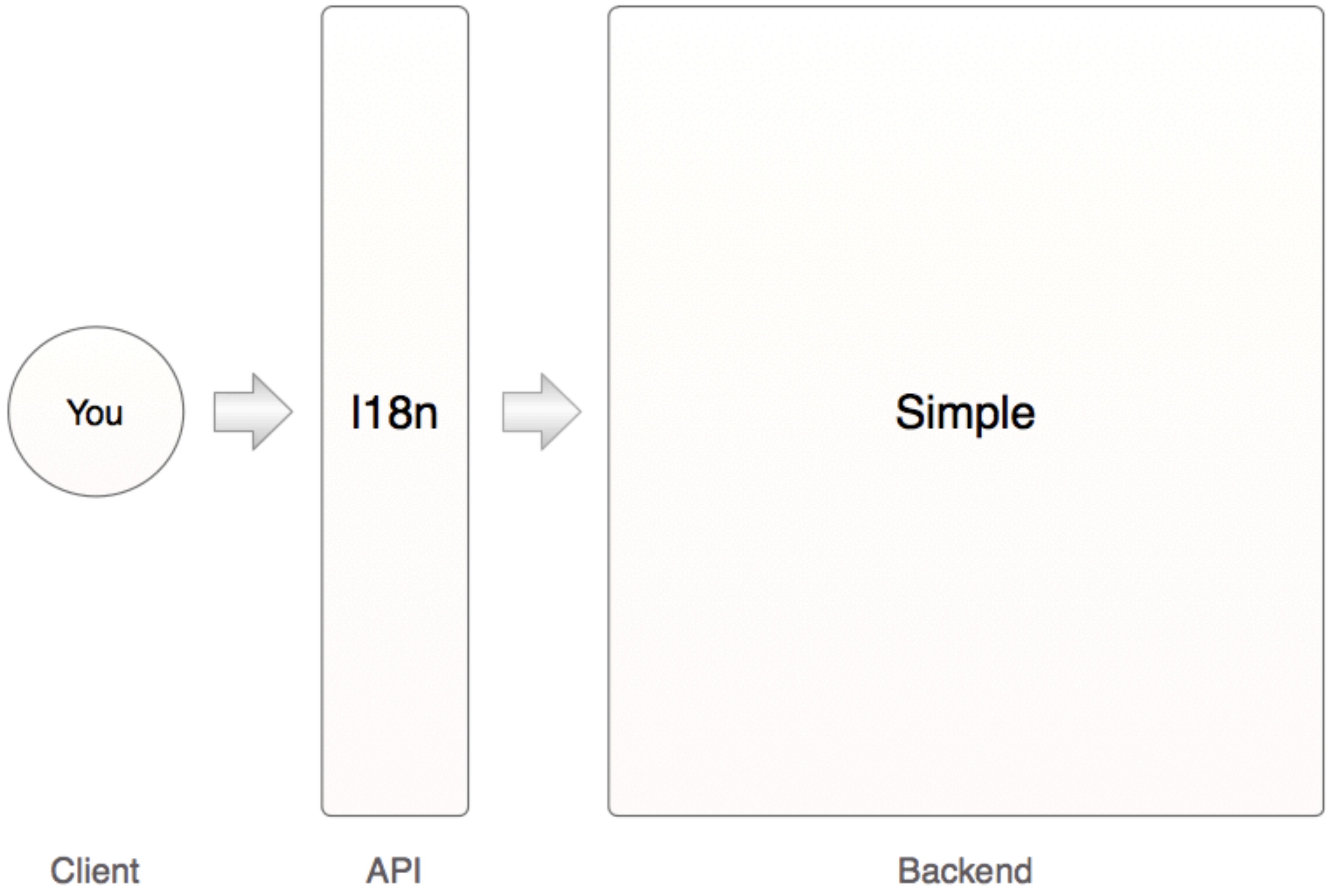
<u>How</u> to solve?

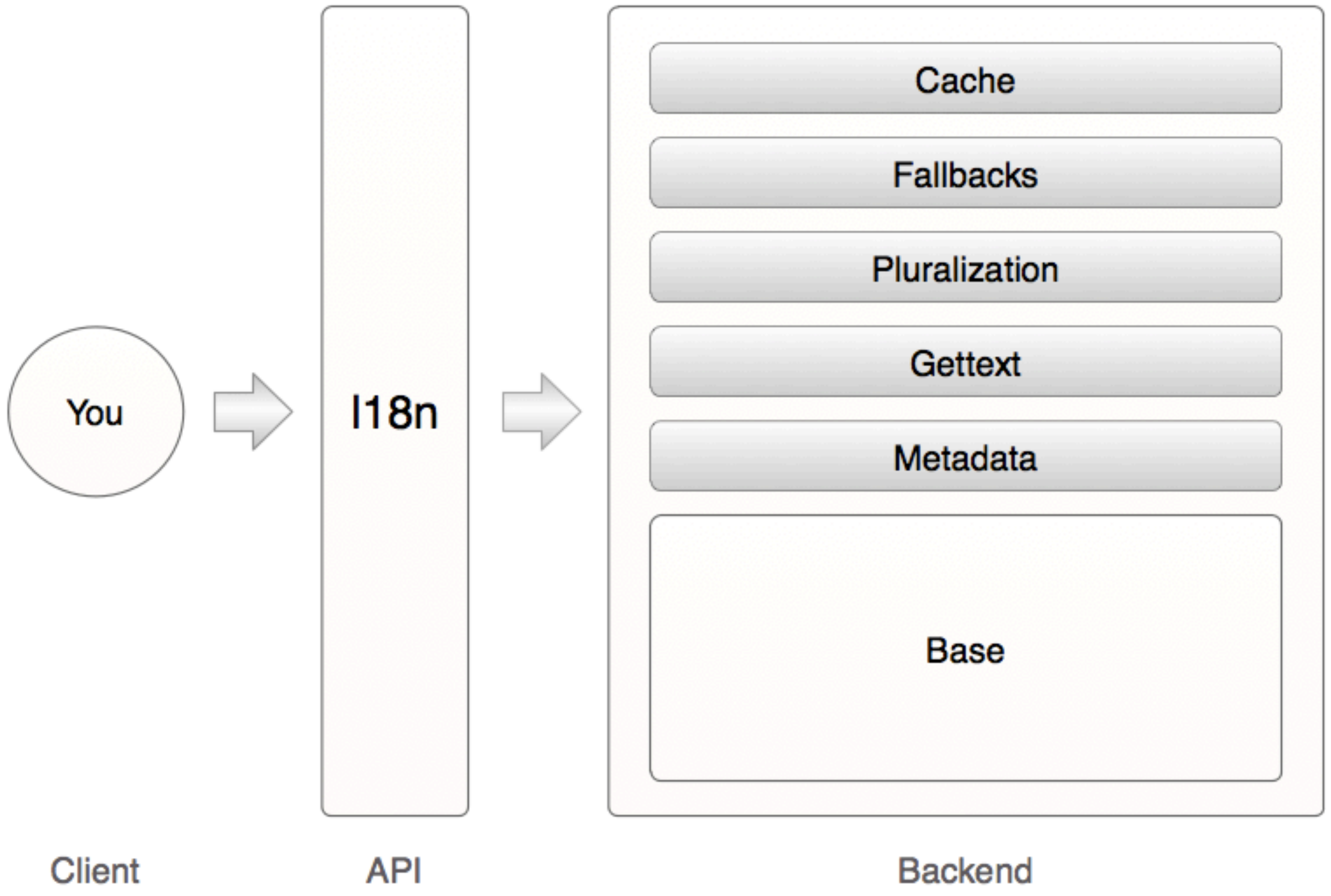
Requirements

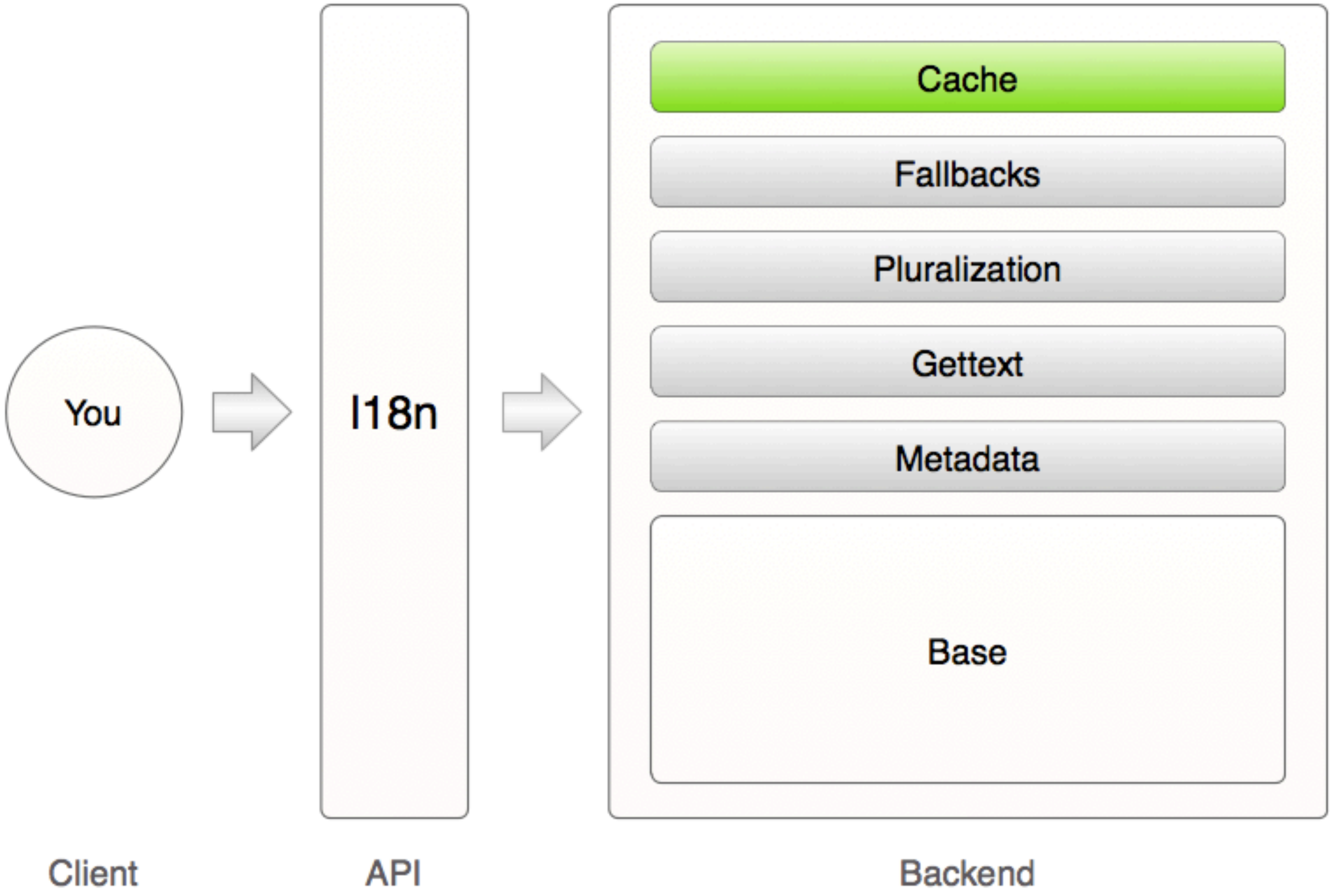
- Simplest thing that possibly could work
- Easy to use
- <u>Very</u> easy to extend

Optional Modules

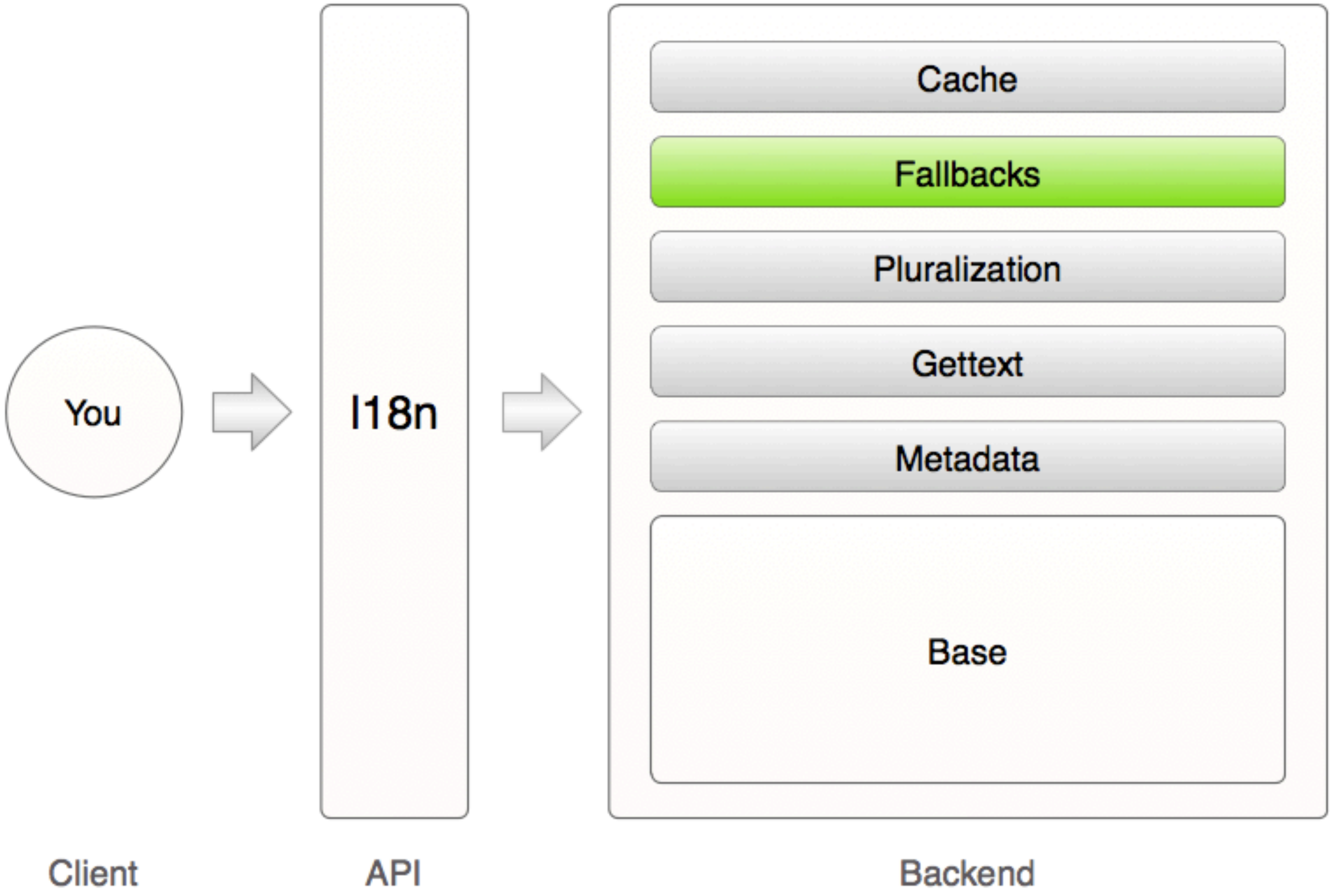
What’s in the box?


You
I18n
Simple
Client
API
Backend


You
I18n
Cache
Fallbacks
Pluralization
Gettext
Metadata
Base
Client
API
Backend

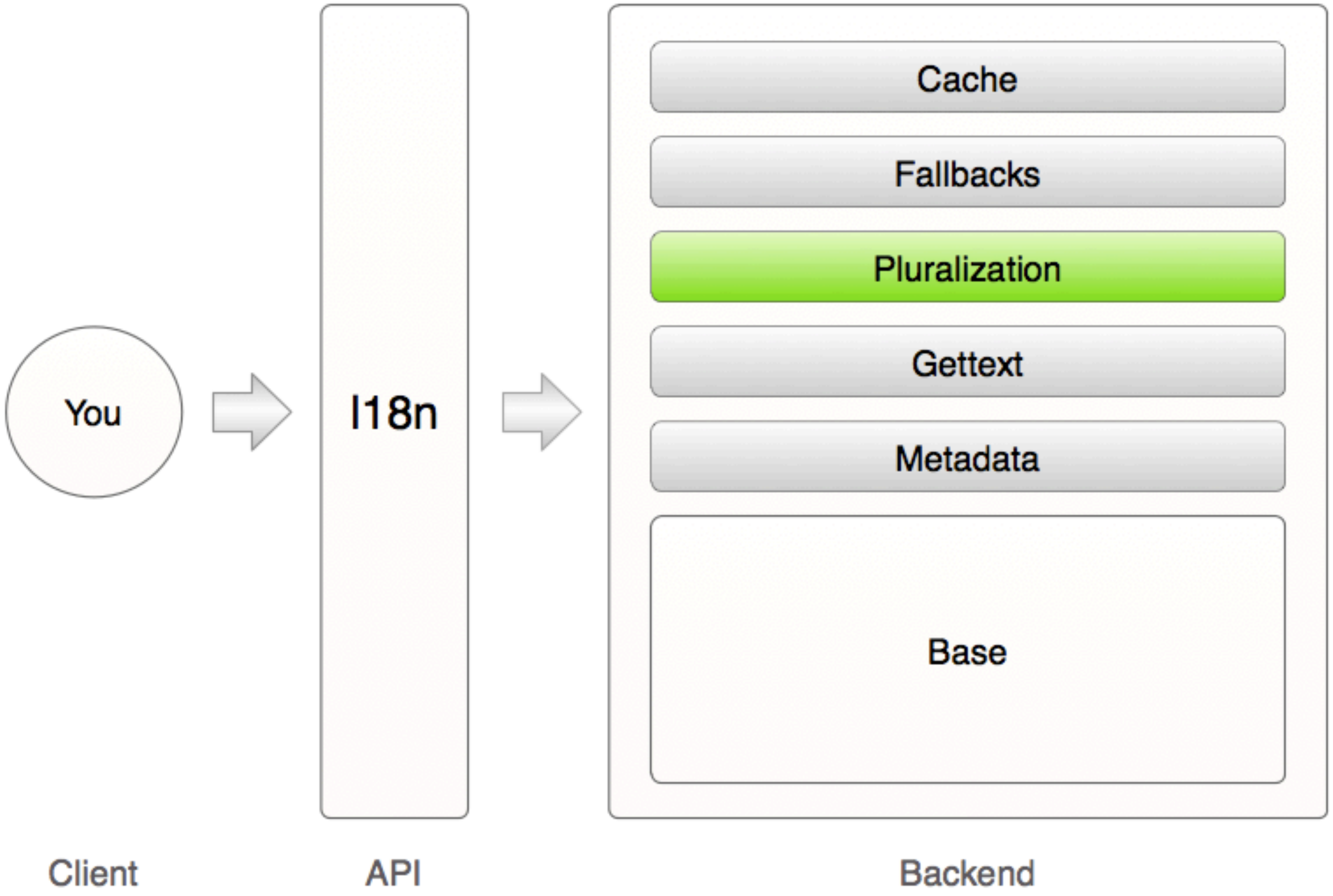

You
I18n
Cache
Fallbacks
Pluralization
Gettext
Metadata
Base
Client
API
Backend

```
I18n.cache_store =
  ActiveSupport::Cache.lookup_store(:memory_store)

# simplified

module Cache
  def translate(*args)
    fetch(*args) { super }
  end

  protected
    def fetch(*args, &block)
      key = cache_key(*args)
      I18n.cache_store.fetch(key, &block)
    end
end
```


You
I18n
Cache
Fallbacks
Pluralization
Gettext
Metadata
Base
Client
API
Backend

```
# when a translation is missing for the current
# locale return the translation for the default
# locale

I18n.fallbacks[:fr] # => [:fr, :en]
I18n.fallbacks[:de] # => [:de, :en]
```

```ruby
# people speaking Catalan also speak Spanish

fallbacks = I18n.fallbacks
fallbacks.map(:ca => :es)
fallbacks[:ca] # => [:ca, :es, :en]

# people speaking Arabian as spoken in Palestine
# also speak Hebrew as spoken in Israel

fallbacks.map(:"ar-PS" => :"he-IL")
fallbacks[:"ar-PS"]
# => [:"ar-PS", :ar, :"he-IL", :he, :en]

# don't fall back to Hebrew for Arabians living
# anywhere else though!

fallbacks[:"ar-EG"]
# => [:"ar-EG", :ar, :en]
```


You
I18n
Cache
Fallbacks
Pluralization
Gettext
Metadata
Base
Client
API
Backend

```ruby
# :en
:message => {
  :one   => "one message"
  :other => "{{count}} messages"
}

key = count == 1 ? :one : :other
```

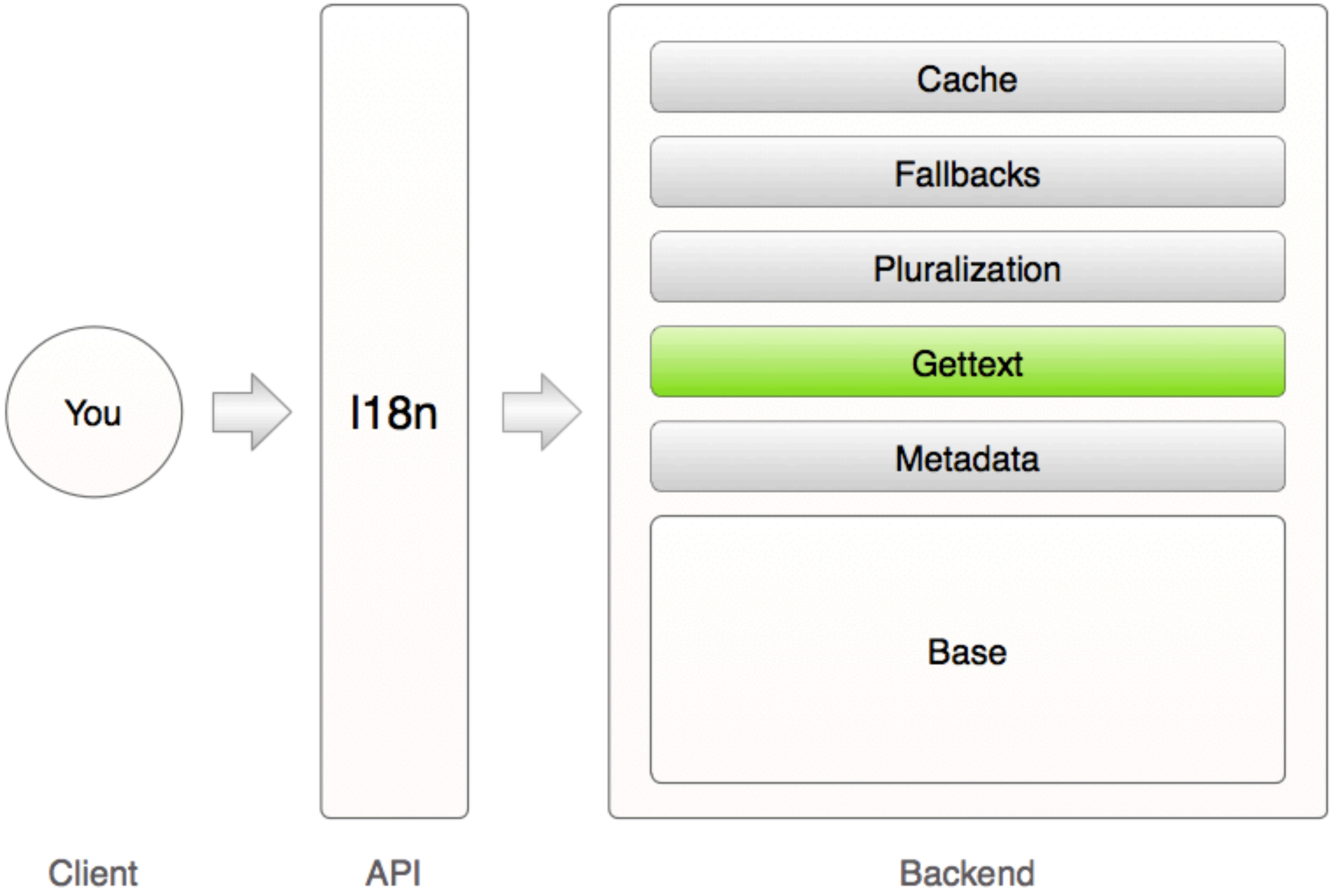
```
# :de
:message => {
  :one   => "Eine Nachricht"
  :other => "{{count}} Nachrichten"
}
```

```
# cs-CZ
:message => {
  :one   => "jedna zpráva"       # 1
  :few   => "{{count}} zprávy" # 2..4
  :other => "{{count}} zpráv"  # >= 5
}
```

```ruby
I18n.locale = :en

t(:message, :count => 1) # => one message
t(:message, :count => 2) # => 2 messages
t(:message, :count => 6) # => 6 messages

I18n.locale = :cs

t(:message, :count => 1) # => jedna zpráva
t(:message, :count => 2) # => 2 zprávy
t(:message, :count => 6) # => 6 zpráv
```

```
:ru => { :i18n => { :plural => { :rule =>

  lambda { |n|
    n % 10 == 1 && n % 100 != 11 ? :one :
    [2, 3, 4].include?(n % 10) &&
    ![12, 13, 14].include?(n % 100) ? :few :
    n % 10 == 0 ||
    [5, 6, 7, 8, 9].include?(n % 10) ||
    [11, 12, 13, 14].include?(n % 100) ? :many :
    :other
  }

} } }
```

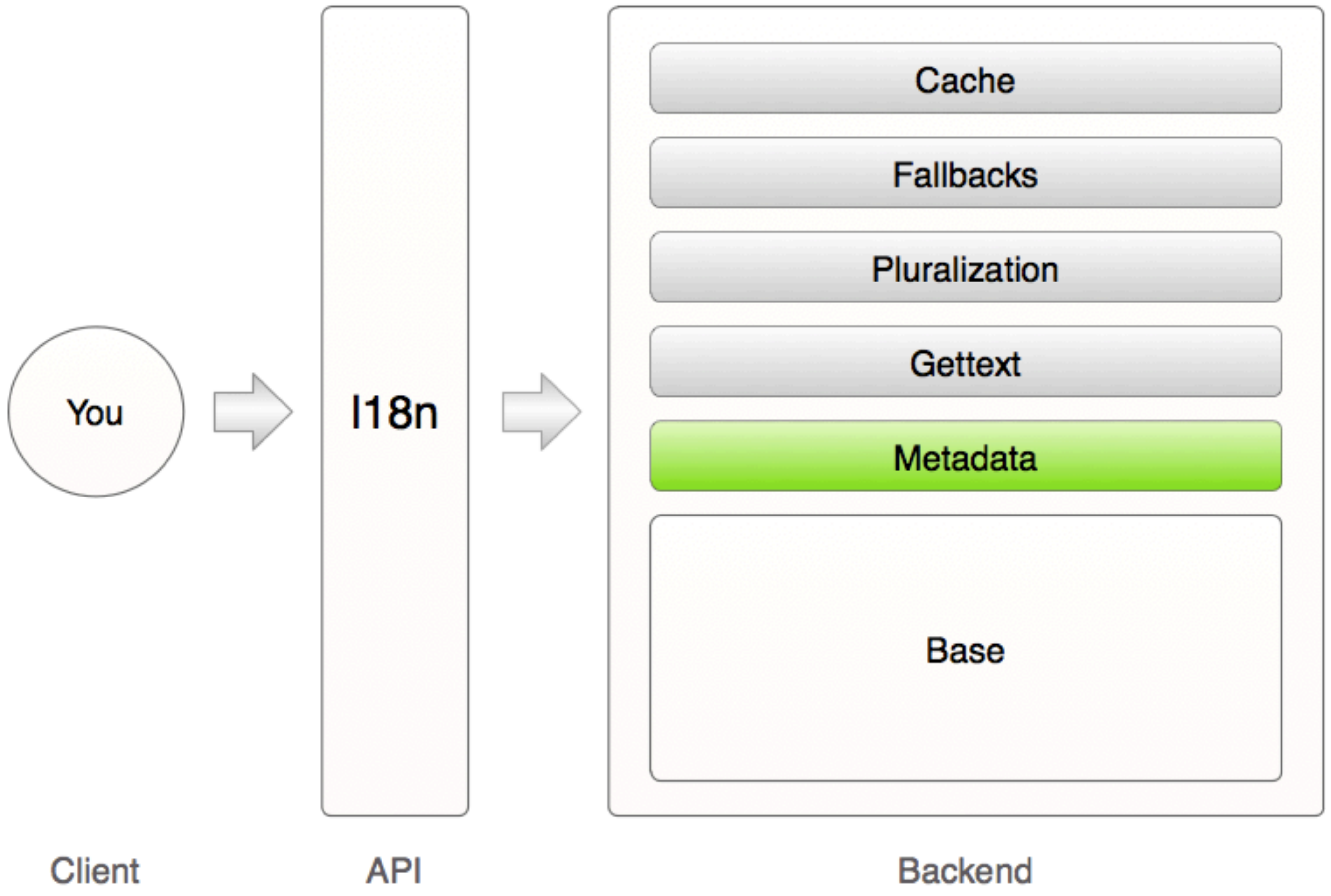
```
lambda { |n|
  # 1, 11, 21 etc. but not 111, 211 etc => :one
  n % 10 == 1 && n % 100 != 11 ? :one :

  # 2, 3, 4, 12, 13, 14 etc.
  # but not 112, 113, 212, 213 etc. => :few
  [2, 3, 4].include?(n % 10) &&
  ![12, 13, 14].include?(n % 100) ? :few :

  # 0, 10, 15, 16 etc.
  # as well as 111, 112 etc. => :many
  n % 10 == 0 ||
  [5, 6, 7, 8, 9].include?(n % 10) ||
  [11, 12, 13, 14].include?(n % 100) ? :many :

  # everything else => :other
  :other
}
```


You
I18n
Cache
Fallbacks
Pluralization
Gettext
Metadata
Base
Client
API
Backend

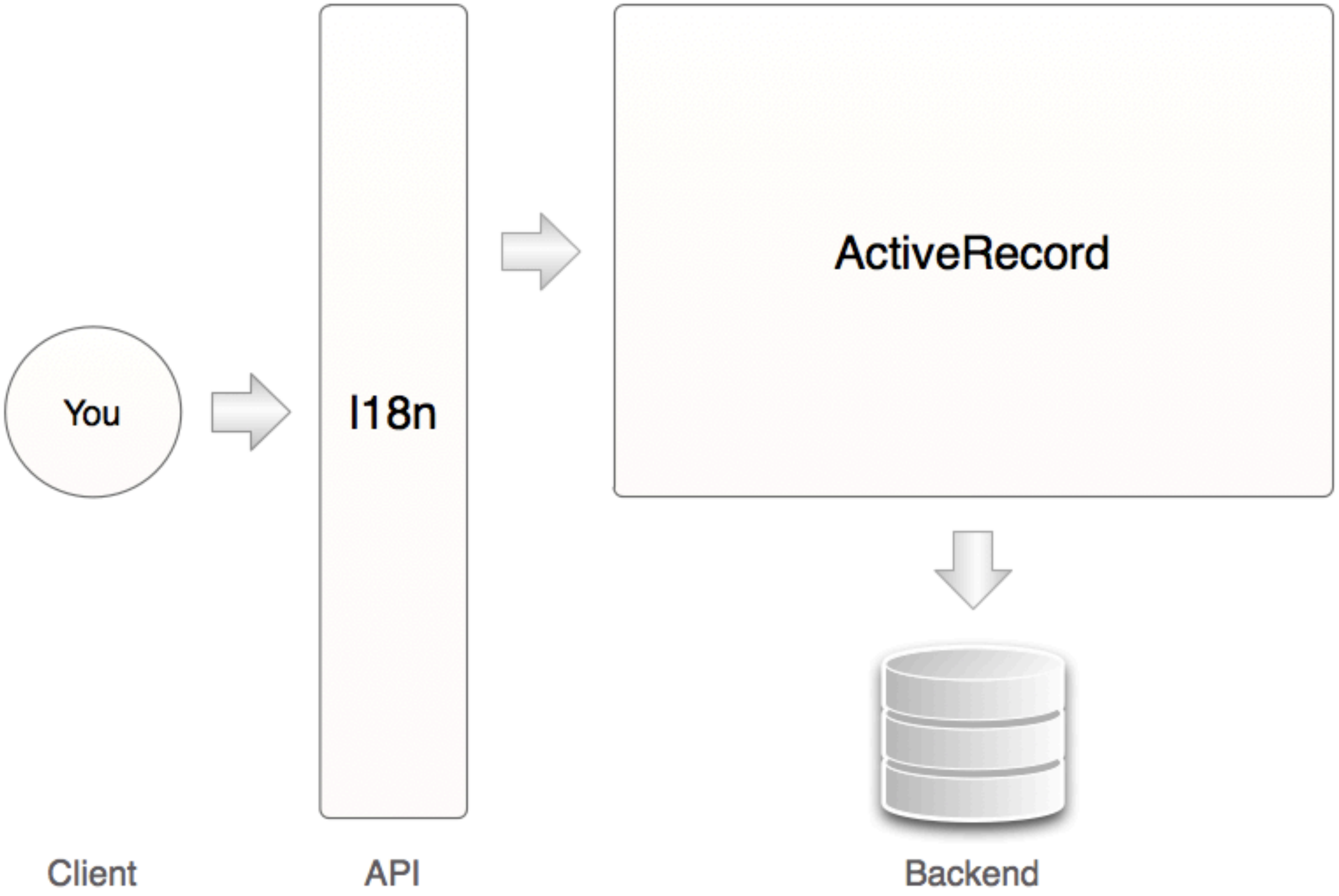
```ruby
class I18n::Backend::Simple
  include I18n::Backend::Gettext
end

I18n.load_path += Dir["path/to/locales/*.po"]

t("Some message from po file")
# => "Translation from po file"
```

```ruby
I18n.locale = :en
_('car')                 # => car
n_('car', 'cars', 20)    # => 20 cars

I18n.locale = :de
_('car')                 # => Auto
n_('car', 'cars', 20)    # => 20 Autos
```

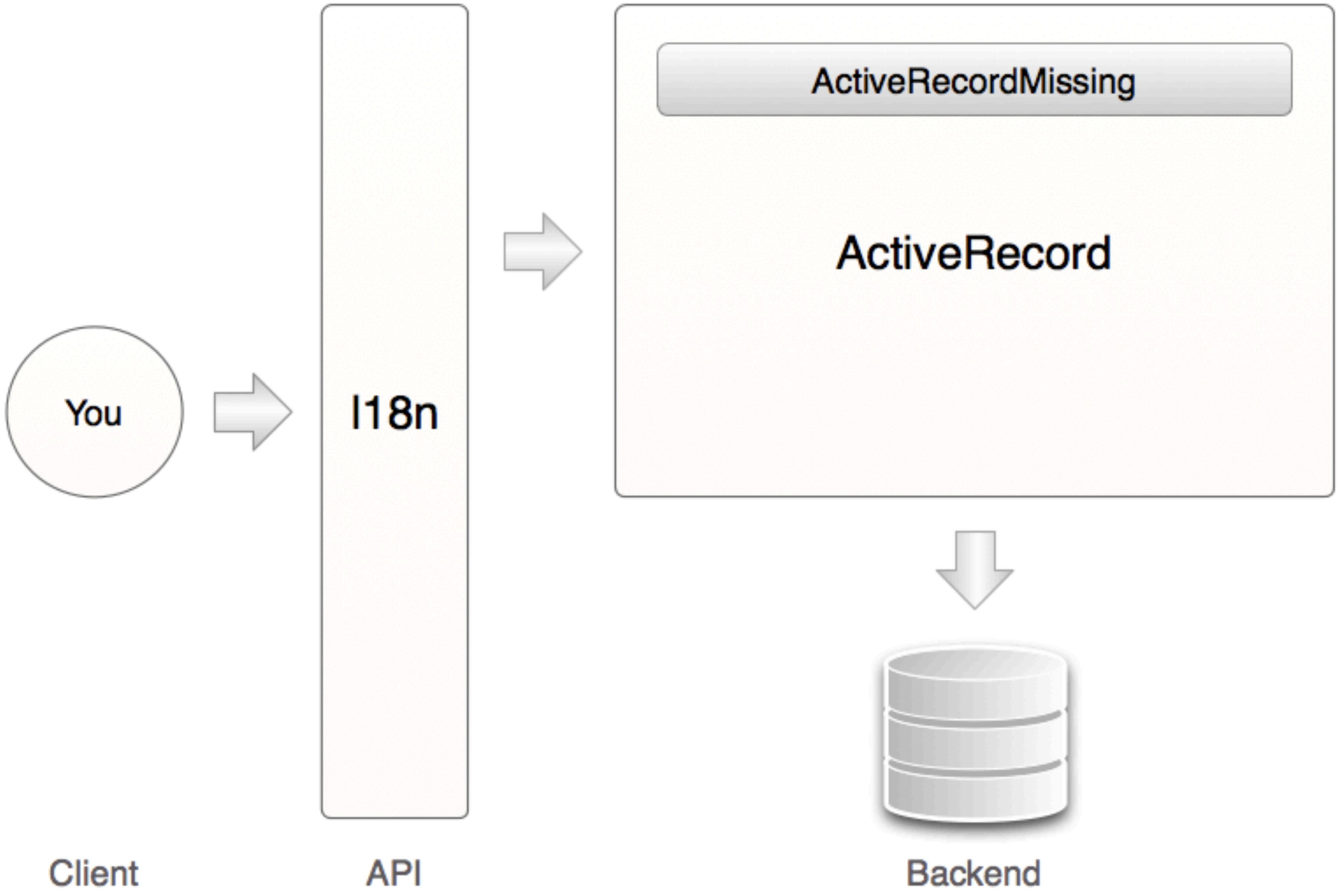
p0wned

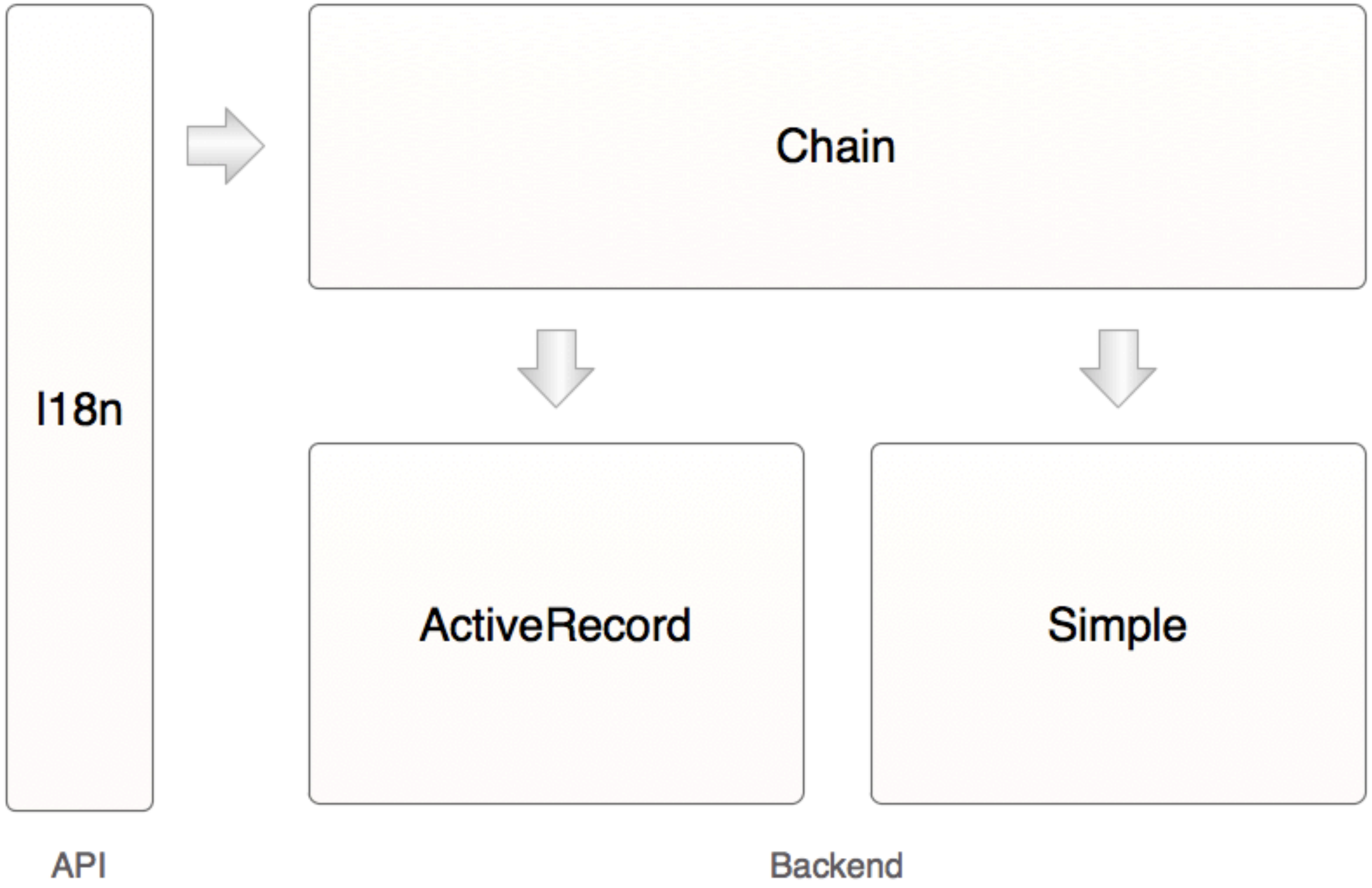

You
I18n
Cache
Fallbacks
Pluralization
Gettext
Metadata
Base
Client
API
Backend

```
# en.yml
:en => {
  :greeting => "Hi {{name}}!"
}

greeting = I18n.t(:greeting, :name => "David")
# => "Hi David!"

greeting.translation_metadata
# => {
#    :locale => :en,
#    :key => :greeting,
#    :original => "Hi {{name}}!",
#    :values => { :name => "David" }
# }
```

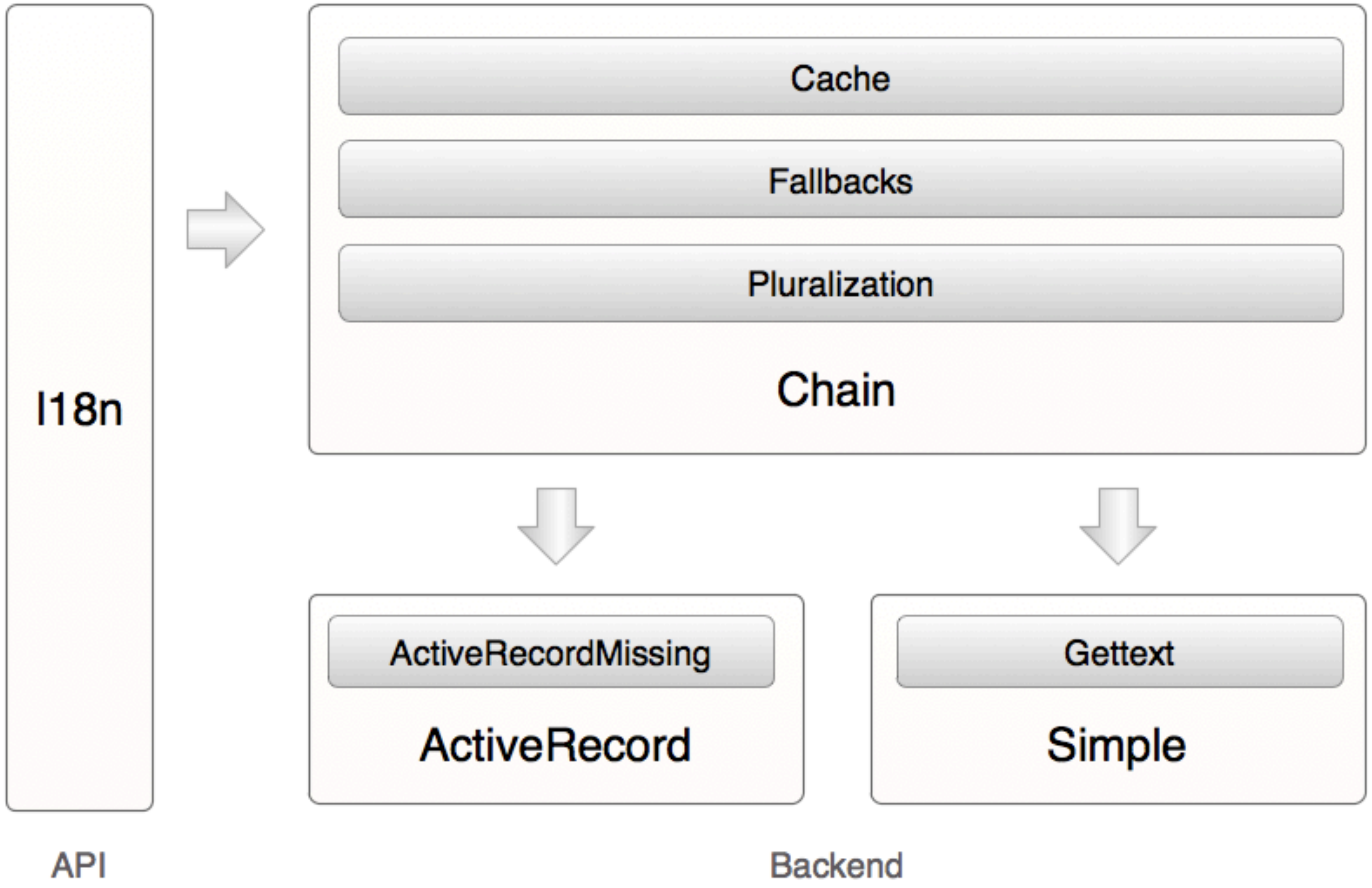

You
I18n
ActiveRecord
Client
API
Backend

```ruby
# assumes you have a translations table in your
# database set up

translations = { :cars => "Cars" }

I18n.backend = I18n::Backend::ActiveRecord.new
I18n.backend.store_translations(:en, translations)

I18n.t(:cars) # => Cars
```


ActiveRecordMissing
ActiveRecord
You
I18n
Client
API
Backend


I18n
Chain
ActiveRecord
Simple
API
Backend

```ruby
first  = I18n::Backend::ActiveRecord.new
first.store_translations(:en, :foo => "FOO")

second = I18n::Backend::Simple.new
second.store_translations(:en, :bar => "BAR")

I18n.backend = \
  I18n::Backend::Chain.new(first, second)

I18n.t(:foo) # => FOO
I18n.t(:bar) # => BAR
I18n.t(:baz) # => raises MissingTranslationData
```


I18n
Cache
Fallbacks
Pluralization
Chain
ActiveRecordMissing
ActiveRecord
Gettext
Simple
API
Backend

Advanced Features

Translation procs

```
# inject logic into the translation lookup process

I18n.locale = :ru
assert_match %r(марта), I18n.l(date, "%d %b %Y")
assert_match %r(март ), I18n.l(date, "%b %Y")
```

```ruby
# en.rb
:en => {
  :greeting => lambda { |values|
    person = values[:person]
    prefix = person.male? ? "Mr." :
             person.married? ? "Mrs." : "Ms."

    "Welcome, #{prefix} #{person.last_name}"
  }
}

I18n.t(:greeting, :person => david)
# => Welcome, Mr. Black!
```

Interpolation procs

experimental

```
# EXPERIMENTAL!

# en.yml
:en => {
  :greeting => "Welcome, {{name}}"
}

name = lambda { |values|
  person = values[:person]
  prefix = person.male? ? "Mr." :
           person.married? ? "Mrs." : "Ms."

  "#{prefix} #{person.last_name}"
}
I18n.t(:greeting, :name => name)
```

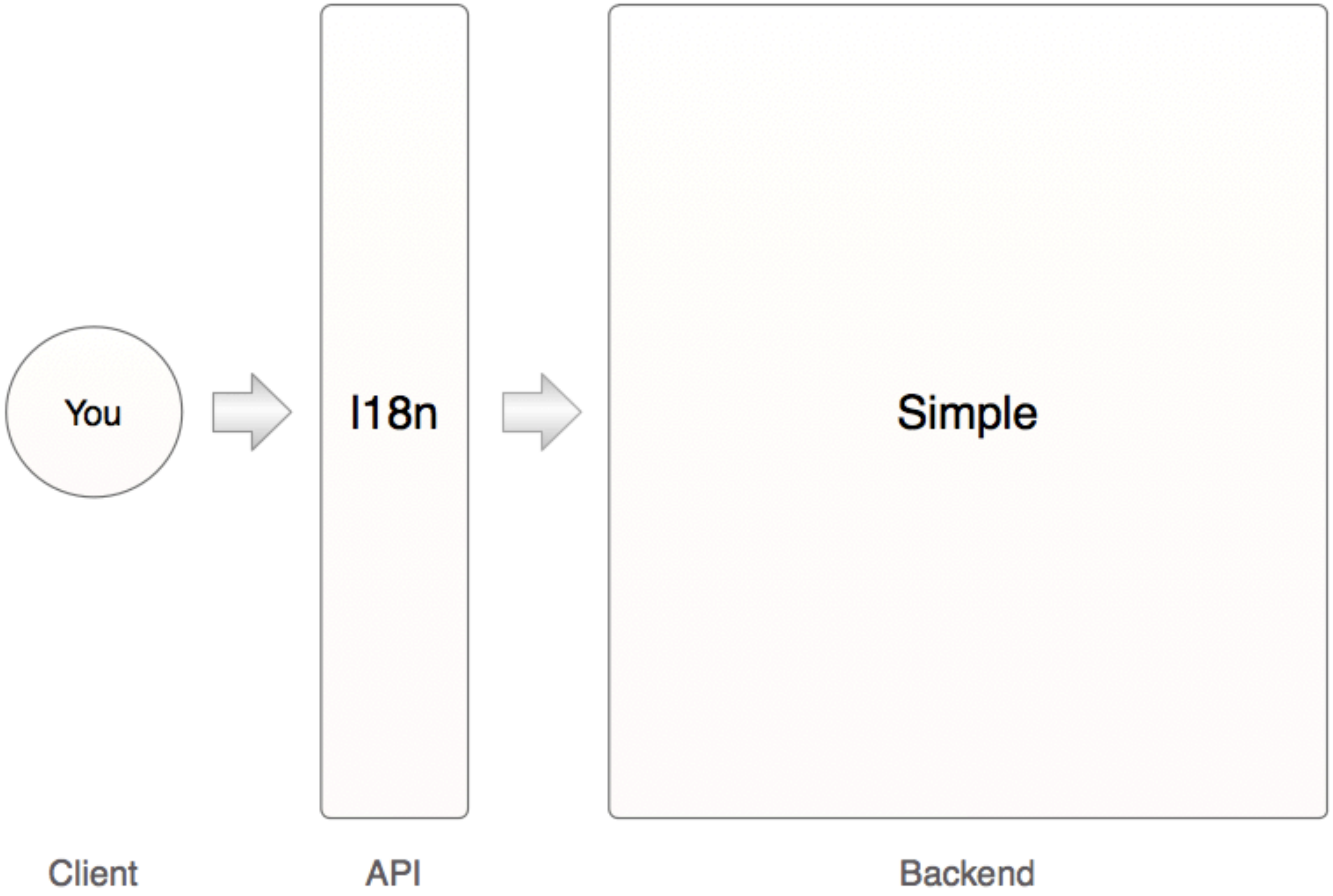
Translation symlinks

experimental

```
# en.yml
en:
  something: Something
  something_else: :something

I18n.t(:something_else) # => Something
```

Library design

The I18n Gem


You
I18n
Simple
Client
API
Backend

```ruby
module I18n                          # simplified
  def backend
    @@backend ||= Backend::Simple.new
  end

  def backend=(backend)
    @@backend = backend
  end

  def translate(key, options = {})
    backend.translate(locale, key, options)
  end
  alias t translate

  class Backend::Simple
    def translate(locale, key, options = {})
      # do the heavy work ...
    end
  end
end
```

How do you extend that?

Let’s simplify ...

Attendee

- Listens (common feature)
- Tweets
- Applauds

```ruby
class Attendee
  def listen
    puts "think: interesting stuff ..."
  end
end

Attendee.new.listen

# think: interesting stuff ...
```

How do you extend that?

```ruby
class Attendee
  def listen
    puts "think: interesting stuff ..."
  end
end

class TweetingAttendee < Attendee
  def listen
    super
    puts "tweet: great conference! :D"
  end
end

TweetingAttendee.new.listen

# think: interesting stuff ...
# tweet: great conference! :D
```

```ruby
class TweetingAttendee < Attendee
  def listen
    super
    puts "tweet: great conference! :D"
  end
end
TweetingAttendee.new.listen

class ApplaudingAttendee < Attendee
  def listen
    super
    puts "applause!"
  end
end
ApplaudingAttendee.new.listen

# now, what’s next?
```

```ruby
class TweetingApplaudingPartyingTShirtifiedAttendee \
  < Attendee
  # WTF
end
```

:-(

```ruby
module ActiveRecord::Locking::Optimistic
  def self.included(base)
    base.alias_method_chain \
      :attributes_from_column_definition, :lock
  end

  def attributes_from_column_definition_with_lock
    # ...
  end
end

class ActiveRecord::Base
  def attributes_from_column_definition
    # ...
  end
end

ActiveRecord::Base.class_eval do
  include ActiveRecord::Locking::Optimistic
end
```

8-}

```ruby
class Attendee
  def listen
    puts "think: interesting stuff ..."
  end
end

module Tweets
  def listen
    super
    puts "tweet: great conference! :D"
  end
end

module Applause
  def listen
    super
    puts "applause!"
  end
end
```

```ruby
attendee = Attendee.new

class << attendee
  include Tweets
  include Applause
end

attendee.listen

# think: interesting stuff ...
# tweet: great conference! :D
# applause!
```

:-)

```ruby
attendee = Attendee.new

class << attendee
  include Tweets
  include Applause
end

attendee.listen

# think: interesting stuff ...
# tweet: great conference! :D
# applause!
```

```ruby
attendee = Attendee.new

class Attendee
  include Tweets
  include Applause
end

attendee.listen
```

```ruby
attendee = Attendee.new

class Attendee
  include Tweets
  include Applause
end

attendee.listen

# think: interesting stuff ...

# http://gist.github.com/244944
```

:-|

```ruby
module Base
  def listen
    puts "think: interesting stuff ..."
  end
end

class Attendee
  include Base
end

module Tweets
  # ...
end

module Applause
  # ...
end
```

```ruby
class Attendee
  include Tweets
  include Applause
end
Attendee.new.listen
```

```ruby
class Attendee
  include Tweets
  include Applause
end
Attendee.new.listen

# think: interesting stuff ...
# tweet: great conference! :D
# applause!

# http://gist.github.com/247648
```

:-D

```ruby
class Attendee
  include Tweets
  include Applause
end
Attendee.new.listen

# think: interesting stuff ...
# tweet: great conference! :D
# applause!

# http://gist.github.com/247648

class << attendee
  include Tweets
  include Applause
end
```

Design Example

I18n Fallbacks & Cache

```ruby
module I18n::Backend::Base
  def translate(locale, key, options = {})
    # do the hard work of implementing the API
  end
end

class I18n::Backend::Simple
  include Base
end

I18n.backend ||= I18n::Backend::Simple.new
```

```ruby
module I18n::Backend::Cache
  def translate(locale, key, options = {})
    # do caching
    super
  end
end

module I18n::Backend::Fallbacks
  def translate(locale, key, options = {})
    # do fallbacks
    super
  end
end
```

```ruby
module I18n::Backend::Cache
  def translate(locale, key, options = {})
    # ...
  end
end

module I18n::Backend::Fallbacks
  def translate(locale, key, options = {})
    # ...
  end
end

class << I18n.backend
  include Cache, Fallbacks
end

class I18n::Backend::Simple
  include Cache, Fallbacks
end
```

Patterns used

Recap

Patterns used

- Swappable backend
- Pluggable modules
- Injectable logic
- Symetry of the API

Thank you!

Questions?

Resources

http://github.com/svenfuchs/i18n

http://guides.rubyonrails.org/i18n.html

http://rails-i18n.org/wiki