SURVIVING APIs WITH RAILS

RESOURCES AND GET

LEVEL 2

IT'S ALL ABOUT THE RESOURCES

Any information that **can be named** can be a resource.

in other words, *nouns*

Some examples of resources:

- A music playlist
- A song
- The leader of the Zombie horde
- Survivors
- Remaining Medical Kits

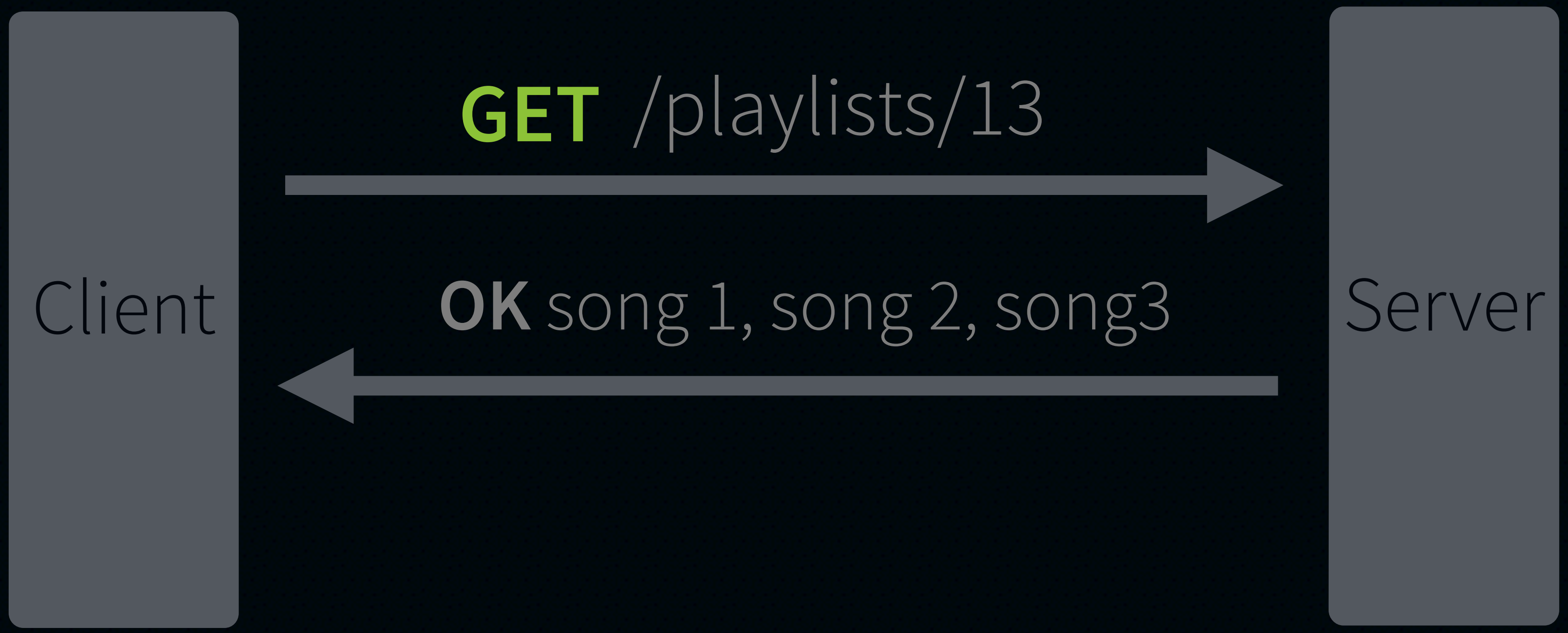
"A resource is a **conceptual mapping** to a set of entities, not the entity that corresponds to the mapping at any particular point in time."

- Steve Klabnik, **Designing Hypermedia APIs**

SURVIVING APIs WITH RAILS

UNDERSTANDING THE GET METHOD

The GET method is used to **read** information identified by a given **URI.**



Important characteristics:

- **Safe** - it should **not** take any action other than retrieval.
- **Idempotent** - sequential **GET** requests to the same URI should not generate side-effects.

WRITING API INTEGRATION TESTS

Integration tests simulate clients interacting with our **API**

config/routes.rb

```ruby
namespace :api, path: '/', constraints: { subdomain: 'api' } do
  resources :zombies
end
```

test/integration/listing_zombies_test.rb

```ruby
require 'test_helper'

class ListingZombiesTest < ActionDispatch::IntegrationTest

  setup { host! 'api.example.com' }

  test 'returns list of all zombies'
end
```

required for testing
with subdomain constraint

WRITING OUR FIRST INTEGRATION TESTS

TEST FAILING

S **200 - Success** status code means the request has succeeded

test/integration/listing_zombies_test.rb

```ruby
require 'test_helper'

class ListingZombiesTest < ActionDispatch::IntegrationTest

  setup { host! 'api.example.com' }

  test 'returns list of all zombies' do
    get '/zombies'
    assert_equal 200, response.status
    refute_empty response.body
  end
end
```

same thing

```ruby
assert response.success?
```

200 responses should include the resource in the response body

SURVIVING APIs WITH RAILS

LISTING RESOURCES

app/controllers/api/zombies_controller.rb

```
module API
  class ZombiesController < ApplicationController
    def index
      zombies = Zombie.all
      render json: zombies, status: 200
    end
  end
end
```

calls the **to_json** method on **zombies**

See the **Rails 4 Patterns** course to learn about **ActiveModel Serializers**

The **to_json** method serializes all properties to JSON

```
zombies.to_json
```

```
[{"id":5,"name":"Joanna","age":null,"created_at":"2014-01-17T18:40:40.195Z",
"updated_at":"2014-01-17T18:40:40.195Z","weapon":"axe"},
{"id":6,"name":"John","age":null,"created_at":"2014-01-17T18:40:40.218Z",
"updated_at":"2014-01-17T18:40:40.218Z","weapon":"shotgun"}]
```

```
zombie.to_json
```

```
{"id":5,"name":"Joanna","age":null,"created_at":"2014-01-17T18:40:40.195Z",
"updated_at":"2014-01-17T18:40:40.195Z","weapon":"axe"}
```

PATH SEGMENTED EXPANSION

Arguments in the URI are separated using a **slash.**

```
/zombies
/zombies/:id
/zombies/:id/victims
/zombies/:id/victims/:id
```

```
/zombies?id=1
```

← this routes to Zombies#index and **NOT** to Zombies#show

MOST URIS WILL NOT DEPEND ON QUERY STRINGS

Sometimes it's ok to use query strings on URIs.

```
/zombies?weapon=axe
```
filters

```
/zombies?keyword=john
```
searches

```
/zombies?page=2&per_page=25
```
pagination

TEST LISTING RESOURCES WITH QUERY STRINGS

TEST FAILING

test/integration/listing_zombies_test.rb

```ruby
class ListingZombiesTest < ActionDispatch::IntegrationTest
  setup { host! 'api.example.com' }

  test 'returns zombies filtered by weapon' do
    john = Zombie.create!(name: 'John', weapon: 'axe')
    joanna = Zombie.create!(name: 'Joanna', weapon: 'shotgun')

    get '/zombies?weapon=axe'
    assert_equal 200, response.status

    zombies = JSON.parse(response.body, symbolize_names: true)
    names = zombies.collect { |z| z[:name] }
    assert_includes names, 'John'
    refute_includes names, 'Joanna'
  end
end
```

if creation logic gets too verbose, use **fixtures** or **FactoryGirl.**

```ruby
{'id' => 51, 'name' => "John"}
```

```ruby
{:id => 51, :name => "John"}
```

LISTING RESOURCES WITH FILTER

TEST PASSING

app/controllers/api/zombies_controller.rb

```
module API
  class ZombiesController < ApplicationController

    def index
      zombies = Zombie.all
      if weapon = params[:weapon]
        zombies = zombies.where(weapon: weapon)
      end
      render json: zombies, status: 200
    end

  end
end
```

Starting in Rails 4, this returns a **chainable scope**

we can add filters dynamically

SURVIVING APIs WITH RAILS

TEST RETRIEVING ONE ZOMBIE

TEST FAILING

test/integration/listing_zombies_test.rb

```
class ListingZombiesTest < ActionDispatch::IntegrationTest
  setup { host! 'api.example.com' }

  test 'returns zombie by id' do
    zombie = Zombie.create!(name: 'Joanna', weapon: 'axe')
    get "/zombies/#{zombie.id}"
    assert_equal 200, response.status

    zombie_response = JSON.parse(response.body, symbolize_names: true)
    assert_equal zombie.name, zombie_response[:name]
  end
end
```

routes to **Zombies#show**

RETURNING ONE ZOMBIE

TEST PASSING

The **:status** option accepts either numbers or symbols.

app/controllers/api/zombies_controller.rb

```
module API
  class ZombiesController < ApplicationController
    def show
      zombie = Zombie.find(params[:id])
      render json: zombie, status: 200
    end
  end
end
```

same thing

```
render json: zombie, status: :ok
```

Visit http://guides.rubyonrails.org/layouts_and_rendering.html
for a list of all numeric status codes and symbols supported by Rails.

SURVIVING APIs WITH RAILS

LOOKS LIKE WE HAVE SOME DUPLICATION

test/integration/listing_zombies_test.rb

```
class ListingZombiesTest < ActionDispatch::IntegrationTest
  setup { host! 'api.example.com' }

  test 'returns zombie by id' do
    zombie = Zombie.create!(name: 'Joanna', weapon: 'axe')
    get "/zombies/#{zombie.id}"
    assert_equal 200, response.status

    zombie_response = JSON.parse(response.body, symbolize_names: true)
    assert_equal zombie.name, zombie_response[:name]
  end
end
```

this method is used multiple times across integration tests

SURVIVING APIs WITH RAILS

USING OUR NEW TEST HELPER

test/integration/listing_zombies_test.rb

```
class ListingZombiesTest < ActionDispatch::IntegrationTest
  setup { host! 'api.example.com' }

  test 'returns zombie by id' do
    zombie = Zombie.create!(name: 'Joanna', weapon: 'axe')
    get "/zombies/#{zombie.id}"
    assert_equal 200, response.status

    zombie_response = json(response.body)
    assert_equal zombie.name, zombie_response[:name]
  end
end
```

EXTRACTING COMMON CODE INTO A TEST HELPER

test/test_helper.rb

```ruby
ENV["RAILS_ENV"] ||= "test"
require File.expand_path('../../config/environment', __FILE__)
require 'rails/test_help'

class ActiveSupport::TestCase
  ActiveRecord::Migration.check_pending!
  fixtures :all

  def json(body)
    JSON.parse(body, symbolize_names: true)
  end
end
```

can be reused across all tests

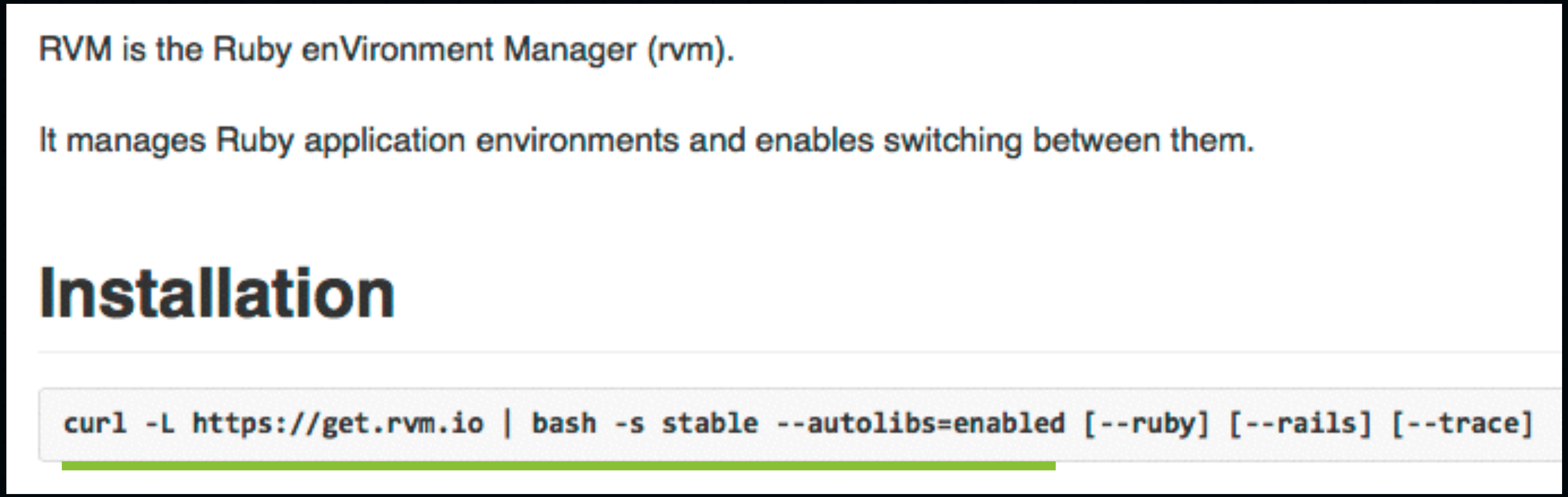
USING CURL TO TEST OUR API WITH REAL NETWORK REQUESTS

curl is a command line tool that issues **real HTTP requests** over the network

defaults to **GET** requests

```
$ curl http://api.cs-zombies-dev.com:3000/zombies
```

A lot of tools use curl as part of their installation process.
For example, **rvm**



RVM is the Ruby enVironment Manager (rvm).

It manages Ruby application environments and enables switching between them.

Installation

```
curl -L https://get.rvm.io | bash -s stable --autolibs=enabled [--ruby] [--rails] [--trace]
```

LOOKING AT THE RESPONSE BODY USING CURL

curl displays the response body on the command line

defaults to **GET** requests

```
$ curl http://api.cs-zombies-dev.com:3000/zombies

[{"id":5,"name":"Joanna","age":null,"created_at":"2014-01-17T18:40:40.195Z",
"updated_at":"2014-01-17T18:40:40.195Z","weapon":"axe"},
{"id":6,"name":"John","age":null,"created_at":"2014-01-17T18:40:40.218Z",
"updated_at":"2014-01-17T18:40:40.218Z","weapon":"shotgun"}]
```

Curl is is shipped with **OS X** and most **GNU/Linux** distributions.
For **Windows** installer, visit http://curl.haxx.se/download.html

USING CURL WITH OPTIONS

works with query strings too!

```
$ curl http://api.cs-zombies-dev.com:3000/zombies?weapon=axe

[{"id":7,"name":"Joanna","age":123,"created_at":"2014-01-17T18:42:47.026Z",
"updated_at":"2014-01-17T18:42:47.026Z","weapon":"axe"}]
```

use the **-I** option to only display response headers

```
$ curl -I http://api.cs-zombies-dev.com:3000/zombies/7

HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-UA-Compatible: chrome=1
Content-Type: application/json; charset=utf-8
...
```