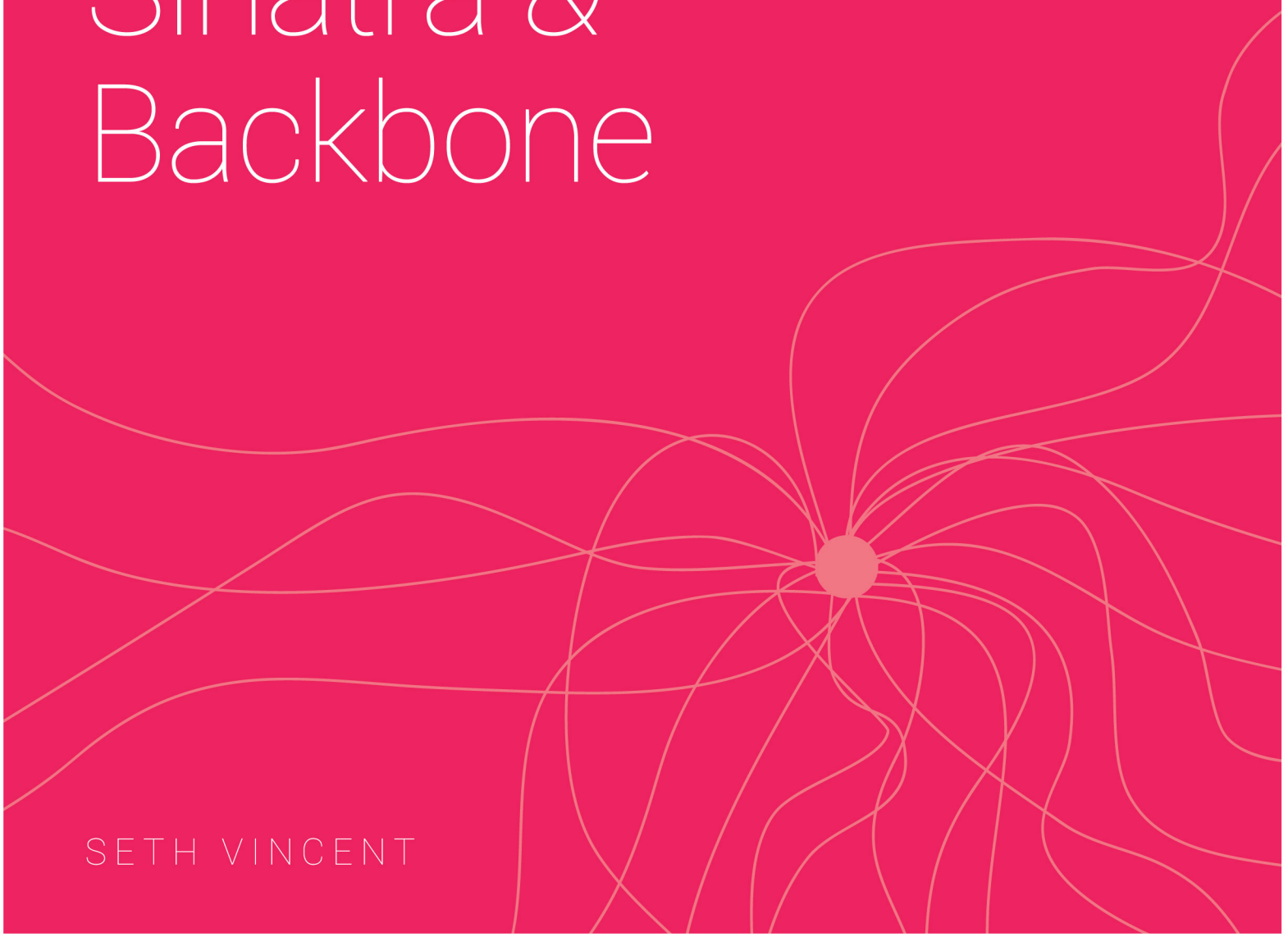
BUILD A BLOG

with

Sinatra & Backbone



SETH VINCENT

Build a blog with Sinatra & Backbone

Create a JSON API with Sinatra, and a front-end client with Backbone.js

Seth Vincent

Contents

Introduction

We're going to make a blog, and we're going to use sinatra and backbone as our tools.

Sinatra is a small ruby framework for building web applications that run on the server.

Backbone is a small javascript framework for building web applications that run in the browser.

Using them together we can create simple, flexible web applications.

Our goal is to make a blog that can have multiple front

servers, clients, and APIs

In this project we'll be developing an API (application programming interface) using sinatra on the server-side, and creating a client-side app using backbone.

So what's it mean for something to be server-side or client-side? And what is an API?

An overview of APIs

Generally, an API is a public interface to a library of code or set of data resources. Libraryies like jQuery, Sinatra, and Backbone expose functionality as APIs. Web apps like Flickr, Twitter and Tumblr expose data as APIs.

With our blog we'll be creating an API that shares articles, tags, and users, so our API will be sharing data rather than code.

We will be exposing articles, tags, and users as resources that client-side apps can consume.

Our server-side app does the job of defining our resources, and essentially creating feeds of the data, endpoints that clients can subscribe to and use.

What's the difference between server and client?

In web development we use the term server-side to specify that the code is run on a server. Client-side typically means that the code runs in a browser.

We'll create one server application using Sinatra that exposes an API, and we might have multiple clients consuming the data from our API.

git, github, and heroku

We'll use git to track changes to our code.

Installing git.

Create an account on GitHub if you haven't already.

git init

git add test.txt

git rm test.txt

git add .

git commit -m 'initial commit'

Create a new repository on github.com

Create an account on heroku.com.

Install heroku toolbelt.

heroku create

git push heroku master

Now, every time you make changes, run:

git add .

git commit -m 'message about your changes'

to push your changes to github: git push origin master

to push your changes to heroku: git push heroku master

Sinatra

Here's a quick preview of a Sinatra app:

```
require 'sinatra'

get '/' do
  { message: 'hello world' }.to_json
end
```

JSON

JSON is a format for data that is often used for web APIs. Flickr, Twitter, and other web services use JSON as the default format for their APIs.

Backbone

Testing with minitest and mocha

minitest https://github.com/seattlerb/minitest

mocha https://github.com/visionmedia/mocha

Throughout this book we will write tests for our app before we write the functionality of our app.

Getting set up

Gem dependencies

Create a Gemfile with this contents:

```
source 'https://rubygems.org'

gem 'sinatra'
gem 'mongoid'
gem 'mongoid_auto_increment_id'

group :test do
  gem 'minitest'
end
```

Create a config.ru file (this will run your app):

```
require 'bundler'
require './blog.rb'

map '/api/v1/' do
 run Blog.new
end
```

Here we're using `map '/api/v1/article/'` to serve our API at that base path.

Now, create blog.rb:

```
require 'rubygems'
require 'sinatra/base'
require 'mongoid'
require './article.rb'

class Blog < Sinatra::Base

  Mongoid.load!('./config/mongoid.yml')

  get '/', :provides => :json do
    {
      title: 'First blog post!',
      content: 'This is a blog post and I like it it is great this is fun.'
    }.to_json
  end

end
```

This code just gets us started. So far we're not really interacting with the database yet, but it gives you a sense of what it takes to send json as a response to a get request. And note that while this says `get '/'`, our app will respond to get requests at `/api/v1/` because of how we mapped the requests for this app.

Articles

Creating the article model in sinatra

Create the article.rb file.

```
class Article
  include Mongoid::Document
  include Mongoid::Timestamps
  include Mongoid::Slug

  field :title, type: String
  field :content, type: String
  field :main_image_url, type: String
  slug :title

  has_and_belongs_to_many :tags
  has_and_belongs_to_many :users
end
```

Creating the article model, collection, and views in backbone

Users

Creating the user model in sinatra

To authenticate our users, we'll employ a ruby gem that eases integration with google: google-api-ruby-client[1]. With this, we'll be able to authenticate users using their google accounts.

Creating the user model, collection and views in backbone

[1] https://github.com/google/google-api-ruby-client

Tags

Creating the tag model in sinatra

Creating the tag model, collection and views in backbone

Site configuration

Creating the site configuration model in sinatra

Creating the site configuration model and views in backbone

Styling the blog

Administering the blog

Deploying the client-side code

jsonp

GitHub pages

Phonegap

Embedding your blog and its posts