Introduction to Web Development

using Sinatra

Konstantinos Karasavvas

CITY College

November 13, 2013

Table of contents

Migrations

- A database schema continuously evolves
- Synchronising with other devs or dev machines becomes important
- Typically, changes are captured with an SQL script
- Migrations provide:
  - a Ruby DSL to describe modifications
  - a framework to run multiple migrations to upgrade to downgrade your database
- Example: Tiny Blog
  - two tables: `authors` and `posts`
  - a one to many relationship
- The first migration will create the database
  - existing databases could be automatically converted to a migration

Migrations: Example

```
# 001_init_db.rb
Sequel.migration do
  up do
    create_table(:authors) do
      primary_key :id
      String :name
    end

    create_table(:posts) do
      primary_key :id
      String :body

      # many to one with authors
      foreign_key :author_id, :authors
    end
  end

  down do
    drop_table(:posts)
    drop_table(:authors)
  end
end
```

Migrations: Example, cont

```
$ sequel -m db/migrations sqlite://db/tblog.db
```

```
$ sqlite3 db/tblog.db
SQLite version 3.7.9 2011-11-01 00:52:41
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
authors      posts        schema_info
sqlite> .schema authors
CREATE TABLE `authors`(`id` integer DEFAULT (NULL) NOT NULL ↩
    PRIMARY KEY, `name` varchar(255) DEFAULT (NULL) NULL);
sqlite>
```

Migrations: Example, cont

```
# 002_add_author_age.rb
Sequel.migration do
  change do
    alter_table(:authors) do
      add_column :age, Integer
    end
  end
end
```

```
$ sequel -E -m db/migrations sqlite://db/tblog.db
...
```

Migrations: sequel -E ...

```
I, [2013-08-03T12:24:13.163269 #6904]  INFO -- : (0.000425s) PRAGMA foreign_keys = 1
I, [2013-08-03T12:24:13.163771 #6904]  INFO -- : (0.000077s) PRAGMA case_sensitive_like = 1
I, [2013-08-03T12:24:13.169883 #6904]  INFO -- : (0.000316s) SELECT NULL FROM `schema_info` LIMIT 1
I, [2013-08-03T12:24:13.170723 #6904]  INFO -- : (0.000303s) SELECT * FROM `schema_info` LIMIT 1
I, [2013-08-03T12:24:13.171488 #6904]  INFO -- : (0.000231s) SELECT 1 AS 'one' FROM `schema_info` LIMIT 1
I, [2013-08-03T12:24:13.172428 #6904]  INFO -- : (0.000255s) SELECT COUNT(*) AS 'count' FROM `schema_info` LIMIT 1
I, [2013-08-03T12:24:13.173217 #6904]  INFO -- : (0.000240s) SELECT `version` FROM `schema_info` LIMIT 1
I, [2013-08-03T12:24:13.174023 #6904]  INFO -- : Begin applying migration version 2, direction: up
I, [2013-08-03T12:24:13.187388 #6904]  INFO -- : (0.000211s) SELECT sqlite_version() LIMIT 1
I, [2013-08-03T12:24:13.187812 #6904]  INFO -- : (0.000112s) PRAGMA foreign_keys
I, [2013-08-03T12:24:13.188009 #6904]  INFO -- : (0.000059s) PRAGMA foreign_keys = off
I, [2013-08-03T12:24:13.188230 #6904]  INFO -- : (0.000057s) BEGIN
I, [2013-08-03T12:24:13.189121 #6904]  INFO -- : (0.000602s) ALTER TABLE `authors` ADD COLUMN `age` integer
I, [2013-08-03T12:24:13.320536 #6904]  INFO -- : (0.131190s) COMMIT
I, [2013-08-03T12:24:13.320836 #6904]  INFO -- : (0.000080s) PRAGMA foreign_keys = on
I, [2013-08-03T12:24:13.454859 #6904]  INFO -- : (0.133744s) UPDATE `schema_info` SET `version` = 2
I, [2013-08-03T12:24:13.464961 #6904]  INFO -- : Finished applying migration version 2, direction: up, took 0.290906 seconds
```

Migrations: Example, cont

```
$ sqlite3 db/tblog.db
SQLite version 3.7.9 2011-11-01 00:52:41
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
authors      posts        schema_info
sqlite> .schema authors
CREATE TABLE `authors`(`id` integer DEFAULT (NULL) NOT NULL ↩
    PRIMARY KEY, `name` varchar(255) DEFAULT (NULL) NULL, `age` ↩
    integer);
sqlite>
```

```
$ sequel -m db/migrations -M 1 sqlite://db/tblog.db
```

```
$ sequel -m db/migrations -M 0 sqlite://db/tblog.db
```

Associations

- Relationships between tables (models) are called associations
  - Example: Tiny Blog
  - Authors has a one to many relationship with posts
- Sequel provides an easy way to declare
  - one to many
  - many to one
  - many to many

Sequel Associations: Example

```
# encoding: utf-8
class Author < Sequel::Model
  one_to_many :posts
end
```

```
# encoding: utf-8
class Post < Sequel::Model
  many_to_one :author
end
```

- `many_to_many` is as simple as the above example
  - ... but assumes a correct schema (i.e. three tables)

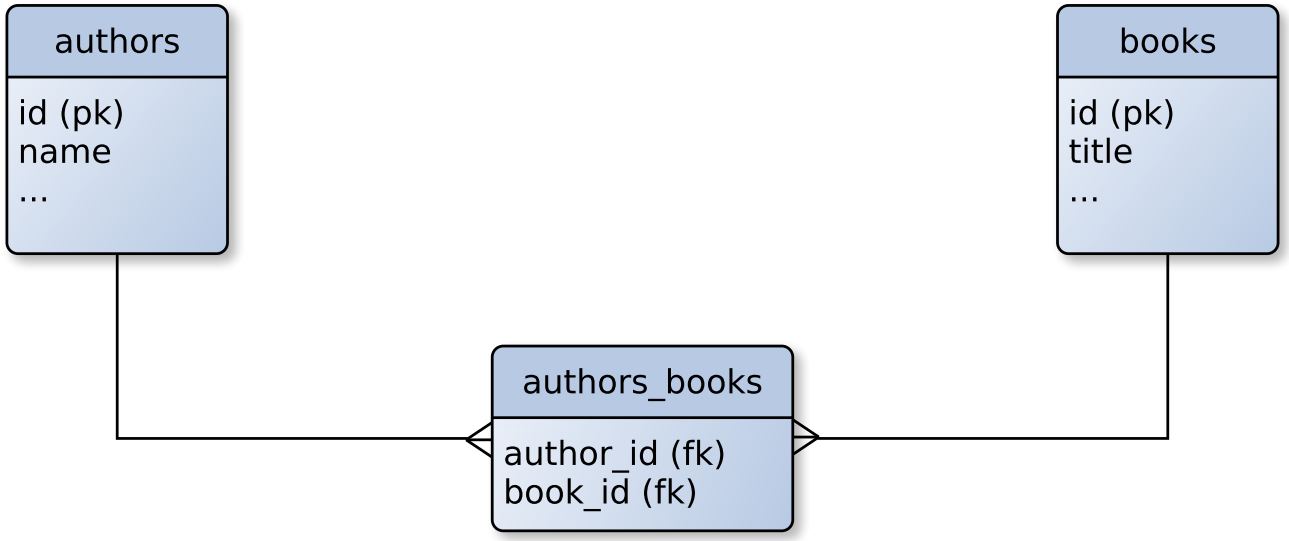
Sequel Associations: Example 2

- An author can write many books
- A book can have many authors
- Sequel models will simply be:

```
# encoding: utf-8
class Author < Sequel::Model
  many_to_many :books
end
```

```
# encoding: utf-8
class Book < Sequel::Model
  many_to_many :authors
end
```

Sequel Associations: Example 2, cont.



Sequel Associations: Example 2, cont.

```
books = [
          Book.create(:title => "Ruby"),
          Book.create(:title => "Sinatra")
        ]

author = Author.create(:name => "Kostas Karasavvas")

books.each do |b|
  author.add_book(b)
end

author.save
```

Basic Authentication

- HTTP Basic Authentication
  - simplest authentication possible
  - no cookies, sessions, logins
- Uses standard HTTP headers
- No confidentiality (encryption) of credentials
  - encoded with BASE64
  - no cookies, sessions, logins
- Credentials are cached by the browser to avoid constant prompting
- Server-side
  - HTTP 401 Not Authorized
  - `WWW-Authenticate: Basic realm="insert realm"`
- Client-side
  - concatenate "username:password"
  - `Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==`
- `http://username:password@www.example.com/path`

Basic Authentication: Example 1

```
# basic_auth.rb
class AppAdmin < Sinatra::Application

  use Rack::Auth::Basic, "Admin Area" do |username, password|
    username == 'foo' && password == 'bar'
  end

  get '/' do
    "admin only"
  end

  get '/another' do
    "another admin only"
  end

end

class App < Sinatra::Application
  get '/' do
    "the app"
  end
end
```

Basic Authentication: Example 1, cont.

```
#config.ru
require 'sinatra'
require './basic_auth.rb'

run Rack::URLMap.new({
  "/" => App,
  "/admin" => AppAdmin
})
```

- '/'
- '/admin'
- '/admin/another'

Basic Authentication: Example 2

```
# basic_auth2.rb
class App < Sinatra::Application

  before '/admin/*' do
    protected!
  end

  helpers do
    def protected!
      unless authorized?
        throw(:halt, [401, "Login incorrect\n"])
      end
    end

    def authorized?
      response['WWW-Authenticate'] = %(Basic realm="Admins only")
      @auth ||=  Rack::Auth::Basic::Request.new(request.env)
      @auth.provided? && @auth.basic?  && @auth.credentials && ↩
          @auth.credentials == ['admin', 'admin']
    end
  end
```

Basic Authentication: Example 2, cont.

```
  get '/' do
    "the app"
  end

  get '/admin' do
    "admin only"
  end

  get '/admin/another' do
    "another admin only"
  end

end
```

```
#config2.ru
require 'sinatra'
require './basic_auth2.rb'

run App
```

OAuth

- Open standard for authentication
  - allows clients to access server resources using another service's credentials
  - e.g. login to the app using twitter credentials
  - Facebook, Google, GitHub, FourSquare, Dropbox, ...
- OmniAuth: multiple-provider authentication library
  - implemented as Rack middleware
  - provides OAuth for several providers, called *Strategies*
- OAuth requires registration of the web app with the respective provider
  - a static web app address is needed
  - the provider will then give you a **key** and a **password**

OmniAuth: Example (libs/config)

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

gem 'sinatra', '~> 1.3.2'
gem "haml"
gem "omniauth-twitter"

#gem "omniauth-facebook"
```

```
#config.ru

require 'bundler'
Bundler.require

require './oauth.rb'

run App
```

OmniAuth: Example (app)

```
# oauth.rb
class App < Sinatra::Application

  configure do
    enable :sessions
  end

  use OmniAuth::Builder do
    provider :twitter, ENV["TWITTER_KEY"], ENV["TWITTER_PASS"]
    provider :developer
  end
```

OmniAuth: Example (app), cont.

```
  get '/' do
    haml :welcome
  end

  %w(get post).each do |method|
    send(method, "/auth/:provider/callback") do
      session['username'] = env['omniauth.auth']['info']['email']
      redirect "/"
    end
  end

  get '/auth/failure' do
    # params[:message] contains the authentication error
    # if using sinatra-flash or rack-flash use the following
    flash[:notice] = params[:message]
    redirect '/'
  end

end
```

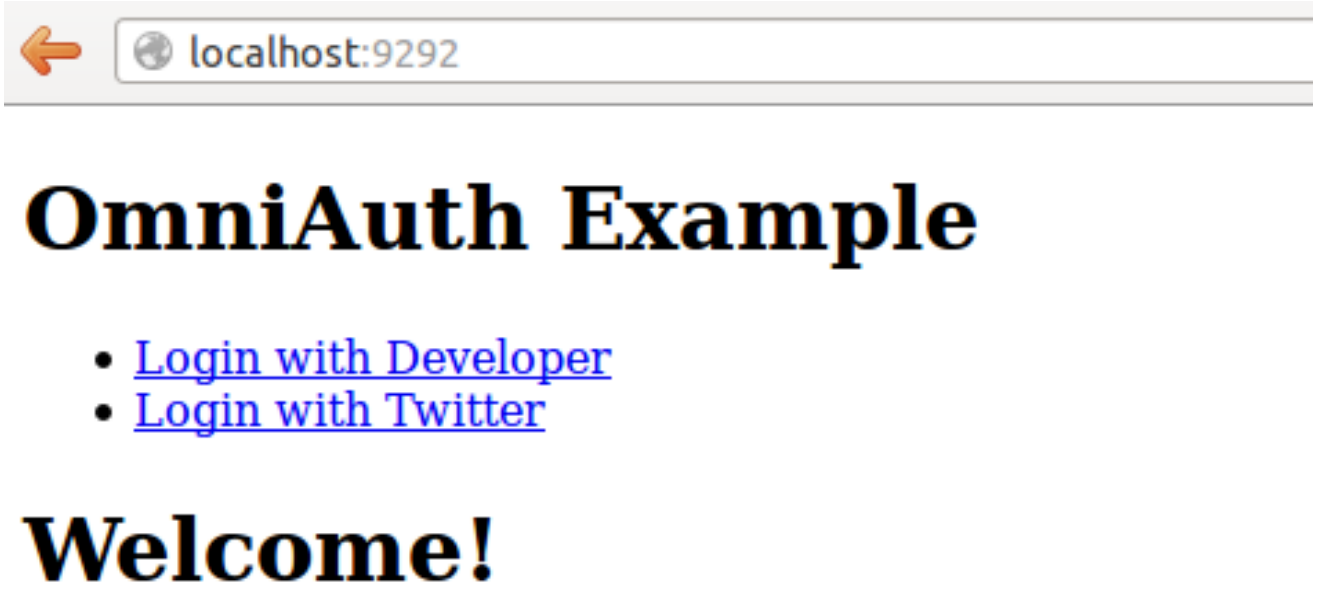
OmniAuth: Example (views)

```
!!! 5
%html
  %head
    %title OmniAuth Example
  %body
    %h1 OmniAuth Example
    -unless session['username']
      %ul
        -if settings.environment == :development
          %li
            %a(href="auth/developer") Login with Developer
        %li
          %a(href="auth/twitter") Login with Twitter
    -else
      %p You are logged in as #{session['username']}

  =yield      # main content
```

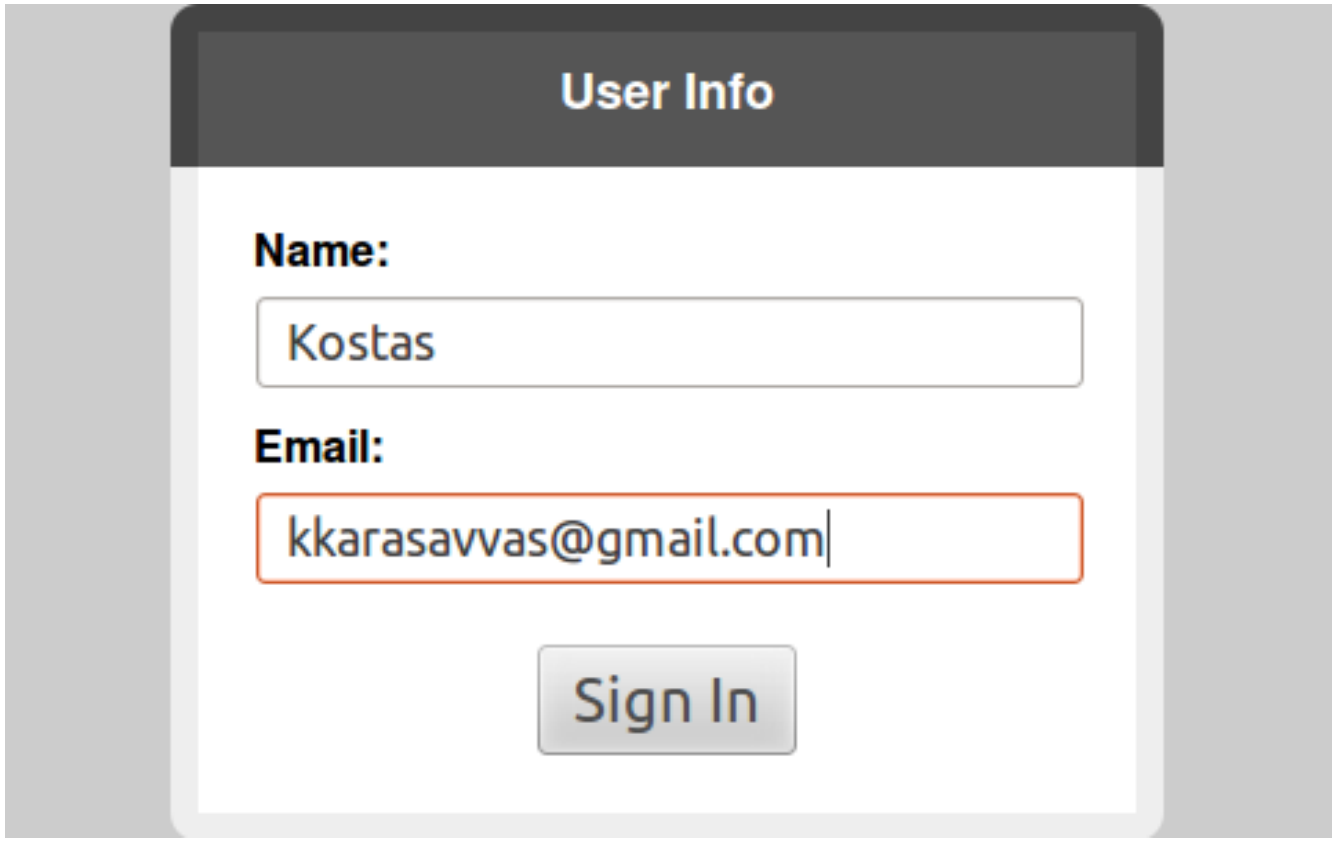
```
-# welcome.haml
%h1 Welcome!
```

OmniAuth: Example (screenshots)



OmniAuth: Example (screenshots, cont.)

- `http://localhost:9292/auth/developer`



OmniAuth: Example (screenshots, cont.)