Guille Carlos @guillec

- Bitpop LLC
- 51 state
- Family Double Dare
- How this will work:
  - I talk, then you code.
  - We will be working with two directories.
- Feedback: TDP
- Anyone not user linux based machine?
- The gob framework


ILLUSIONS
TRICKS ARE WHAT A WHORE DOES FOR MONEY

Building a gem

- Our new framework like Rails will be a gem.
- Gem is a Ruby library that an application can include and build on

Create empty gem (30 sec)

```
> bundle gem gob
     create  gob/Gemfile
     create  gob/Rakefile
     create  gob/.gitignore
     create  gob/gob.gemspec
     create  gob/lib/gob.rb
     create  gub/lib/gob/version.rb
Initializating git repo in src/gob
```

Gob: gemspec file

```
#gob.gemspec
gem.name        = "gob"
gem.version     = Gob::VERSION
gem.authors  = ["Gob Bluth"]
gem.email       = ["gob@thebluthcompany.com"]
gem.homepage    = "www.thebluthcompany.com"
gem.summary     = %q{A Rack-based Web Framework}
gem.description = %q{A Rack-based Web Framework,
                  but with extra hotness}
```

Gob: dependencies (1 min)

#gob.gemspec
gem.add_runtime_dependency "rack"

Rack

- Rack is a gem to interface your framework to a Ruby application server such as Mongrel, Thin, Lightppd, Passenger, WEBrick or Unicorn.

- For now, know that it’s how Ruby turns HTTP requests into code running on your server.

Build and Install Gob (30 sec)

```
> gem build gob.gemspec
> gem install gob-0.0.1.gem
```

Ruby on Gob (1 min)

- Typically we do "rails new your_app"

- We will do it manually:

> mkdir bananas
> cd bananas
> mkdir config
> mkdir app
> touch Gemfile

Ruby on Gob (10 sec)

#bananas/Gemfile

```
source :rubygems
gem 'gob'
```

Install (30 sec)

bundle install

- Gemfile.lock will be created and all dependencies will be installed

- The Gemfile.lock makes your application a single package of both your own code and the third-party code it ran the last time you know for sure that everything worked. (From gembundler.com)

Gob: Add Rack (1 min)

```
#gob/lib/gob.rb

module Gob
  class Application
    def call(env)
      [200, {'Content-Type' => 'text/html'}, ["Come On!"]]
    end
  end
end
```

Build Gob gem again (10 s)

> gem build gob.gemspec
> gem install gob-0.0.1.gem

Bananas App (1 min)

```
#bananas/config/application.rb

require "gob"

module Bananas
  class Application < Gob::Application
  end
end
```

Config.ru (1 min)

#bananas/config.ru

```
require ::File.expand_path('..
/config/application', __FILE__)
run Bananas::Application.new
```

Run Bananas! (1 min)

> rackup -p 3001

- Point your browser to localhost:3001

Building Controller

- How to rout a request like Rails

A little more about Rack

[200, {'Content-Type' => 'text/html'}, ["Come On!"]]

200 -> HTTP Status (404 or 500)

HEADER -> Can return all sorts of other header. This is to tell browser how to render the data.

Come On -> Content

Rack env object

- env is a hash with many values
- The important one for now is PATH_INFO
- PATH_INFO is the part of the URL that is after the server name but before the query params

www.bananas.com**/customer**?q=bluth

Routes

- Based on the URL gob will decided what controller and action to run

- bananas.com**/controller/action**

```
#gob/lib/gob.rb
require "gob/version"
require "gob/routing"
module Gob
  class Application
    def call(env)
      if env['PATH_INFO'] == '/favicon.ico'
        return [404, {'Content-Type' => 'text/html'}, []]
     end

      klass, act = get_controller_and_action(env)
      controller = klass.new(env)
      text = controller.send(act)
      [200, {'Content-Type' => 'text/html'}, [text]]
    end
  end ... CONTINUE ON THE NEXT SLIDE
```

```
  class Controller
    def initialize(env)
      @env = env
    end
    def env
      @env
    end
  end
end
```

Routing.rb

```
# gob/lib/gob/routing.rb
module Gob
  class Application
    def get_controller_and_action(env)
      _, cont, action, after =
        env["PATH_INFO"].split(‘/’, 4)
      cont = cont.capitalize # "People"
      cont += "Controller" # "PeopleController"
      [Object.const_get(cont), action]
    end
  end
end
```

Bananas Controller

```
# bananas/app/controllers/customers_controller.rb
class CustomersController < Gob::Controller
  def show
    "Tobias Funke: The worlds first analyst and therapist."
  end
end
```

require customer controller

```
# bananas/config/application.rb (excerpt)
#Add after "require gob" and before declaring app

$LOAD_PATH << File.join(File.dirname(__FILE__),
                        "..", "app",
                       "controllers")
require "customers_controller"
```

Build Gob

- remember to git add .
- gem build gob.gemspec
- gem install

- Go back to bananas
- rackup -p 3001
- open browser: localhost: 3001/customers/show

The End

- https://github.com/guillec/gob

- Rebuilding Rails.