

# Rails From the Ground Up

**with @nateberkopec**



**Rails is**

**blatant**

**Rails is over. It's a bloated meta framework that requires enormous amounts of peripheral knowledge to understand.**

**— Hacker News**

[Lotus] aims to bring back  
Object Oriented  
Programming to web  
development

— **Lotus web framework**

Lotus is made of  
standalone frameworks  
(controllers, views, etc.)

- **Lotus web framework**

# Lotus is lightweight, fast and testable.

- **Lotus web framework**

# Rails is a massive project

- **Rails: 49,104 commits and 2,551 contributors**
- **Sinatra: 2,664 commits and 225 contributors**
- **Node: 10,222 commits and 548 contributors**
- **Express: 4,974 commits and 165 contributors**
- **Ruby: 38,223 commits and 38 (?) contributors**

**"Rails will become more modular, starting with a rails-core, and including the ability to opt in or out of specific components."**

**— Yehuda Katz, 2008**



All forward progress  
stalled for nearly two  
years, it's *still* slower than  
Rails 2, Bundler is a  
nightmare, Node.js won

— **Jeremy Ashkenas, 2012**

Rails is  
bloated\*

**\*If you let it.**

# What do we get when we use rails new?

- **Empty folders, reminding us where Rails expects to find things**
- **Placeholder files like application.js and application.css, application.html.erb, the application helper and application controller, a locale file, seeds.rb.**

# What do we get when we use rails new?

- **Public folder with a favicon, 404/500 pages, robots.txt**
- **Initializers and config files for different environments**
- **Gemfile**
- **Rakefile**

What do we get when we use rails new *that matters?*

- **config.ru**
- **config/routes.rb**
- **config/application.rb**
- **config/boot.rb**
- **config/environment.rb**

**<http://guides.rubyonrails.org/initialization.html>**

Lets compress five files  
**into one!**

# Gemfile

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

```
gem "rails", "~> 4.2"
```

**This puts a *lot* of stuff in the Gemfile.lock**

# config.ru

```
# normally happens in application.rb via "require 'rails/all'"
require "rails"
require "action_dispatch"
# require "active_controller"
# require "active_record"
# require "action_view"
# require "action_mailer"
# require "active_job"
# require "rails/test_unit"
# require "sprockets"
```



```
# also happens in application.rb
class MyApp < Rails::Application
  # config/routes.rb
  routes.append { root to: Proc.new { [200,[],["Hello world!"]] } }

  config.serve_static_files = false

  # We need a secret token for session, cookies, etc.
  # Usually via config/secrets.yaml
  config.secret_key_base = "insecure"
end
```

```
use Rack::Sendfile
use #<ActiveSupport::Cache:...>
use Rack::Runtime
use Rack::MethodOverride
use ActionDispatch::RequestId
use Rails::Rack::Logger
use ActionDispatch::RemoteIp
use ActionDispatch::Callbacks
use ActionDispatch::Cookies
use ActionDispatch::Session::CookieStore
use ActionDispatch::Flash
use ActionDispatch::ParamsParser
use Rack::Head
use Rack::ConditionalGet
use Rack::ETag
```

```
class MyApp < Rails::Application
  config.middleware.delete ActionController::Cookies
end
```

# Recap

```
# config.ru
require "rails"
require "action_dispatch"

class MyApp < Rails::Application
  routes.append { root to: Proc.new { [200,[],["Hello world!"]] } }
  config.secret_key_base = "insecure"
end
```

```
class HelloController < ActionController::Metal
  include AbstractController::Rendering
  include ActionController::Rendering

  def world
    render text: "Hello world!"
  end
end
```

# ActionController::Metal

- Inherits from **AbstractController::Base**
- Doesn't include a lot of the things you normally get in Rails controllers
- No layouts, no render, no nothin'

# All controllers are also Rack apps

```
class HelloController < ActionController::Metal
  include ActionController::Rendering
  include ActionController::Rendering

  def world
    render text: "Hello world!"
  end
end

run HelloController.action(:world)
# get 'hello', 'hello#index'
# get 'hello', to: HelloController.action(:index)
```

**AbstractController::Rendering,AbstractController::Translation,AbstractController::AssetPaths,Helpers,HideActions,UrlFor,Redirecting,ActionView::Layouts,Rendering,Renderers::All,ConditionalGet,EtagWithTemplateDigest,RackDelegation,Caching,MimeResponds,ImplicitRender,StrongParameters,Cookies,Flash,RequestForgeryProtection,ForceSSL,Streaming,DataStreaming,AbstractController::Callbacks,Rescue,Instrumentation,ParamsWrapper**



# Recap

```
require "rails"
require "action_controller"

class MyApp < Rails::Application
  routes.append { root "hello#world" }
  config.secret_key_base = "insecure"
end

class HelloController < ActionController::Metal
  include ActionController::Rendering
  include ActionController::Rendering

  def world
    render text: "Hello world!"
  end
end
```

```
# config/environment.rb
```

```
MyApp.initialize!
```

```
# config.ru
```

```
run MyApp
```

```
require "rails"
require "action_controller"
class MyApp < Rails::Application
  routes.append { root "hello#world" }
  config.secret_key_base = "insecure"
end
class HelloController < ActionController::Metal
  include AbstractController::Rendering
  include ActionController::Rendering

  def world
    render text: "Hello world!"
  end
end
MyApp.initialize!
run MyApp
```

## What do we get in return?

- Remote IP spoofing protection, timing attack prevention via *ActionDispatch::Remotelp*
- Automatic reloading in development
- Environments
- Excellent logging (*ActionDispatch::RequestId*, *ActionDispatch::DebugExceptions*)
- Parameter parsing via *ActionDispatch::ParamsParser*
- Conditional GET (*Rack::ConditionalGet*)

## What do we get in return?

- **Caching (*Rack::Cache* and *Rack::ETag*)**
- **HEAD requests to GET via *Rack::Head***
- **Resourceful routes**
- **URL generation and URL helpers**
- **Basic, Token, Digest HTTP auth**
- **A great instrumentation API**
- **Generators**
- **Incredibly simple extensibility**
- **Access to the Rails ecosystem (Engines, gems)**

## Memory differences (Thin)

- **40.1 MB lightweight Rails**
- **70.7 MB stock Rails**
- **26.7 MB Sinatra**

**Most of the difference between Rails and Sinatra at this point is ActiveSupport**

## Speed differences from stock Rails on a microbench

- **Lightweight Rails ~10% faster**
- **Ultra Lightweight Rails ~90% faster (remove all middleware, log to stdout)**
- **Sinatra ~100% faster**

***But* these differences are on the order of single-digit milliseconds. App code > Framework code.**

# Why is this modularity interesting?

- **Improves your understanding of Rails internals**
- **Faster and uses less memory**
- **Win arguments with internet haters**
- **Yehuda spent 2 years on it, be grateful**



# Your homework

- **Don't use rails/all**
- **Try starting from a single file the next time you start a Rails app**

## Bonus: tweet-length Rails apps

```
require "rails/all"
run Class.new (Rails::Application) do
  routes.append{root to:proc{[200,{},[]]}}
end.initialize!
```

**This example requires a secrets.yml and gemfile  
(and the gemfile can't contain ActiveRecord)**

## Bonus: tweet-length Rails apps

```
%w[rails rack_test action_controller].map{|r|require r}
run Class.new (Rails::Application) do
  config.secret_key_base=1
  routes.append{root to:proc{[200,{},[]]}}
end.initialize!
```

**This example can be run from a single file!**

# Expanding: ActiveRecord

```
class Article
  extend ActiveRecord::Naming
  extend ActiveRecord::Translation
  include ActiveRecord::Validations
  include ActiveRecord::Conversion
  attr_accessor :id, :name, :content

  def self.all
    @articles ||= []
  end

  ...etc
end
```

## Expanding: ActiveRecord

- **Add config/database.yml**
- **Set up your database**
- **Require ActiveRecord**
- **Add a Rakefile and call**  
`Rails.application.load_tasks`

# Expanding: *ActionView*

```
class HelloController < ActionController::Metal
  include ActionController::Rendering
  include ActionController::Rendering
  include ActionView::Layouts
  append_view_path "#{Rails.root}/app/views"

  def index
    render "hello/index"
  end
end
```

## Expanding: Rails Server

**— Add back bin/rails and you're set**

# Expanding: *ActionMailer*

— **Just require *ActionMailer* and get to it**



## Expanding: Tests

- **You can do tests in-file, or just require the test support (or your favorite test gem) and hop to it**

