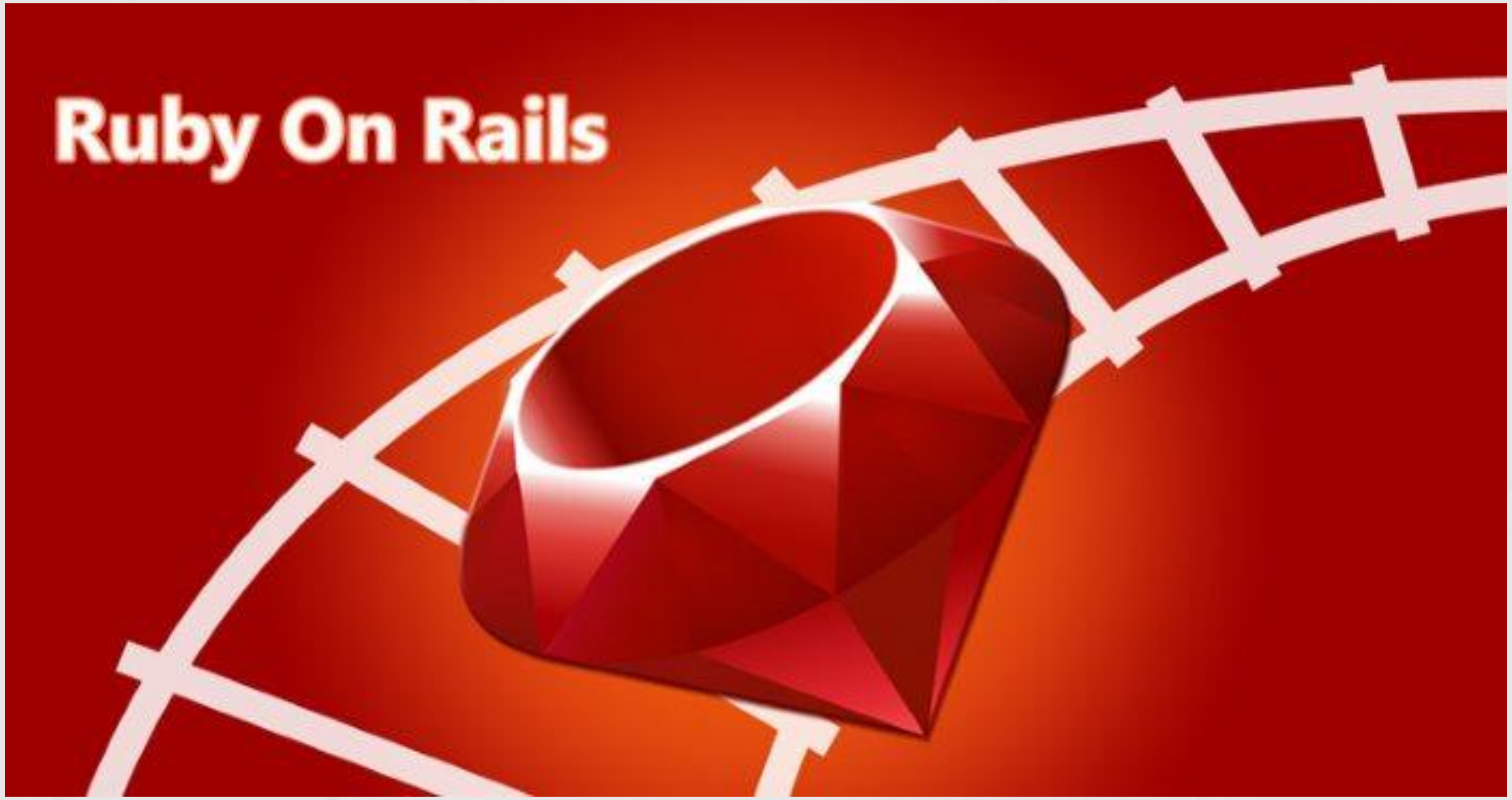
SYNERZIP

Agile Software Product Development Partner



Why Ruby is designed?

SYNERZIP
Agile Software Product Development Partner

Why we should learn Ruby?

- Object Oriented Programing Language
- Scripting Language
- Dynamically Typed
- Duck Type
- Metaprogramming

Data Types

- Integer

  age = 15
- Float

  price = 99.92
- String

  name = “Ruby”
- Boolean

  Is_admin = true
- Array

  array = [15,16.25,’Rails’]
- Hash

  Hash = {:language=>”Ruby”, “framework”:”Rails”}
- Range

  (1..10) => 1 to 10

  (1...10) => 1 to 9

SYNERZIP
Agile Software Product Development Partner

Variables

- Local Variable
  language = ‘Ruby’
- Global Variable
  $language = ‘Ruby’
- Class Variable
  @@language = ‘Ruby’
- Instance Variable
  @language = ‘Ruby’
- CONSTANT
  LANGUAGE= ‘Ruby’

Agile Software Product Development Partner

If else

```
If condition
  Statement
Elsif condition
 Statment
Else
 Statment
end
```

Conditional Operator

```
values  = (num%3 == 0) ? ((1..num).collect{ |i| i*3 }) : (1..num).to_a
```

SYNERZIP
Agile Software Product Development Partner

Loops

- each
- each_with_index
- collect
- select
- reject
- Inject
- Upto
- downto

Each loop

(1..10).each {|i| puts i }

{:language=>”Ruby”, :framework=> “Rails”}.each{|key,value| puts “Key: #{key} and Value #{value}”}

Each_with_index loop

(1..10).each_with_index {|i,index| puts “Value #{val} and index #{index}” }

map loop

Puts (1..10).map {|i| i*2 }

Select loop

Puts (1..10).select {|i| i%2== 0 }

reject loop

Puts (1..10).reject {|i| i%2== 0 }

inject loop

```
Puts (1..10).inject(:+)
```

Upto loop

```
(1).upto(10) {|i| puts i}
```

Downto loop

```
(10).downto(1) {|i| puts i}
```

Class

```
class className
  def initialize
    // statments
  end
  def self.method_name
     // statments
  end
 def method_name
    // statments
 end
end
```

Methods

Class Methods

```
def self.method_name
  // statments
end
```

Instance Methods

```
def method_name
  // statments
end
```

Modules

- Modules are fragment of code we can include it in
  - Other Classes
  - Other Modules
- We can create namespace using Modules
- We can use it for multiple inheritance
- We can not initialize module

Define Modules

```
module ModuleName
  def method_name
    // statements
  end
  def self.method_name
    // statements
  end
end
```

What is Rails?

- Rails is a framework

Who are using?

Basecamp

shopify

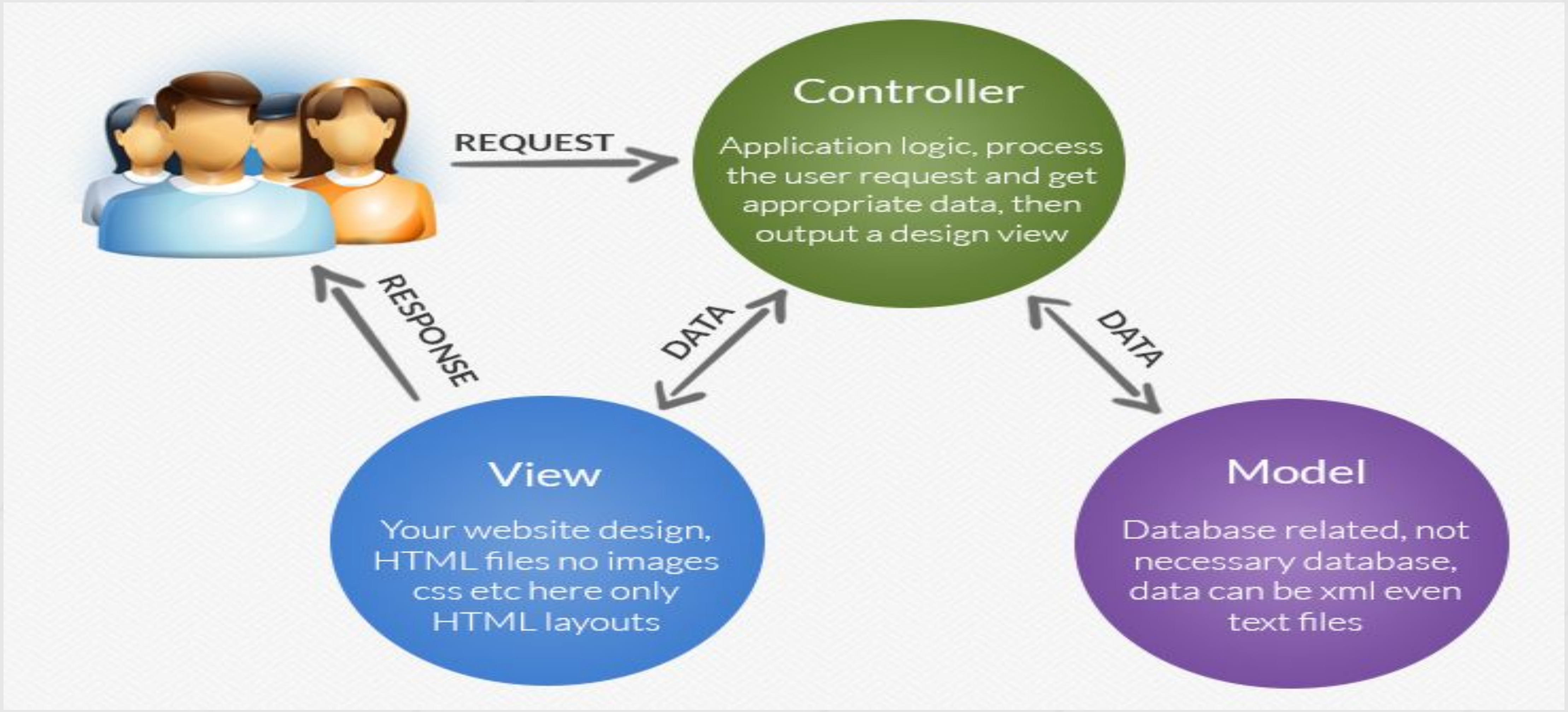
SOUNDCLOUD

airbnb

Why we should learn?

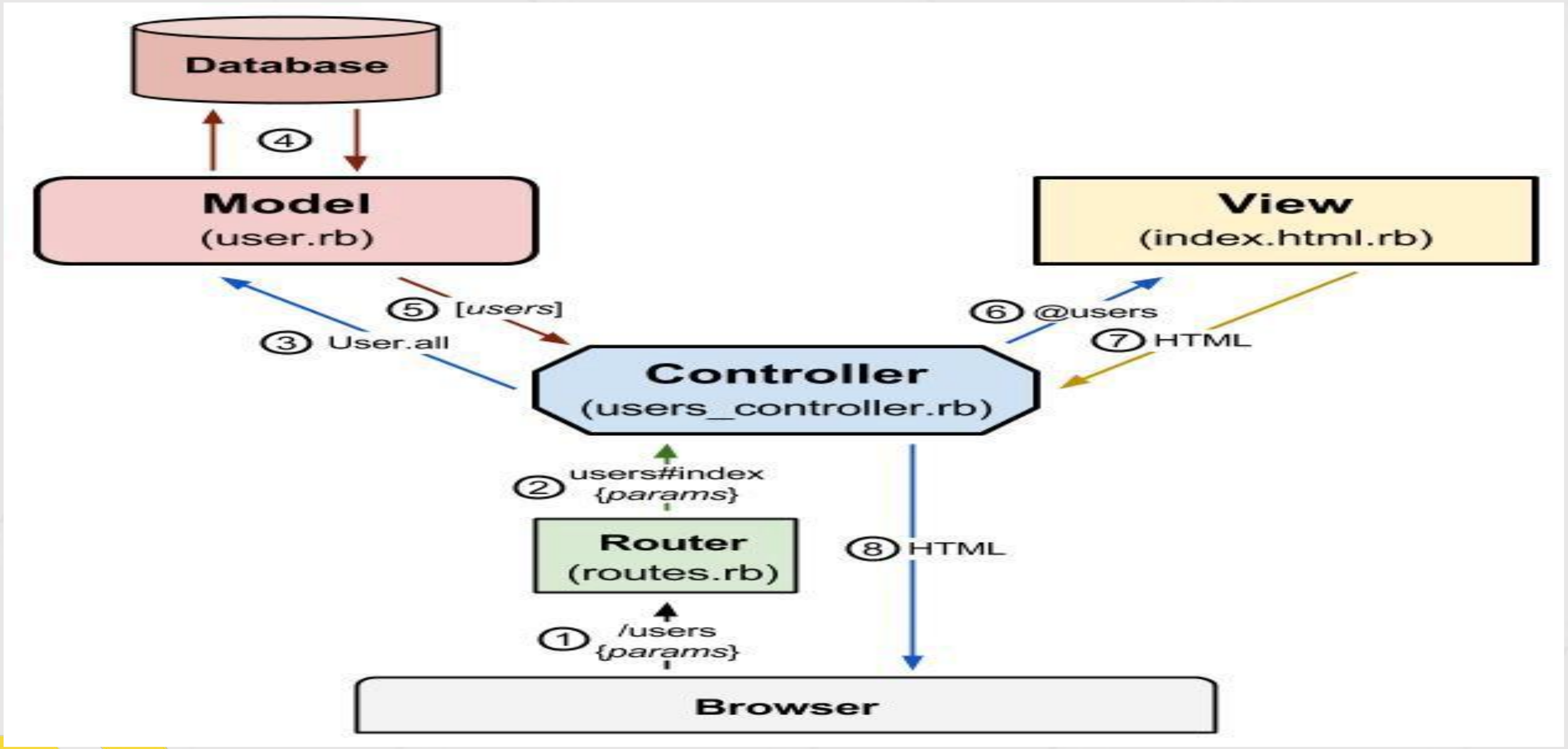
- It encourage DRY Principle
- It work on Convention over Configuration Principle

MVC


REQUEST
Controller
Application logic, process the user request and get appropriate data, then output a design view
RESPONSE
DATA
DATA
View
Your website design, HTML files no images css etc here only HTML layouts
Model
Database related, not necessary database, data can be xml even text files

SYNERZIP
Agile Software Product Development Partner

MVC Architecture



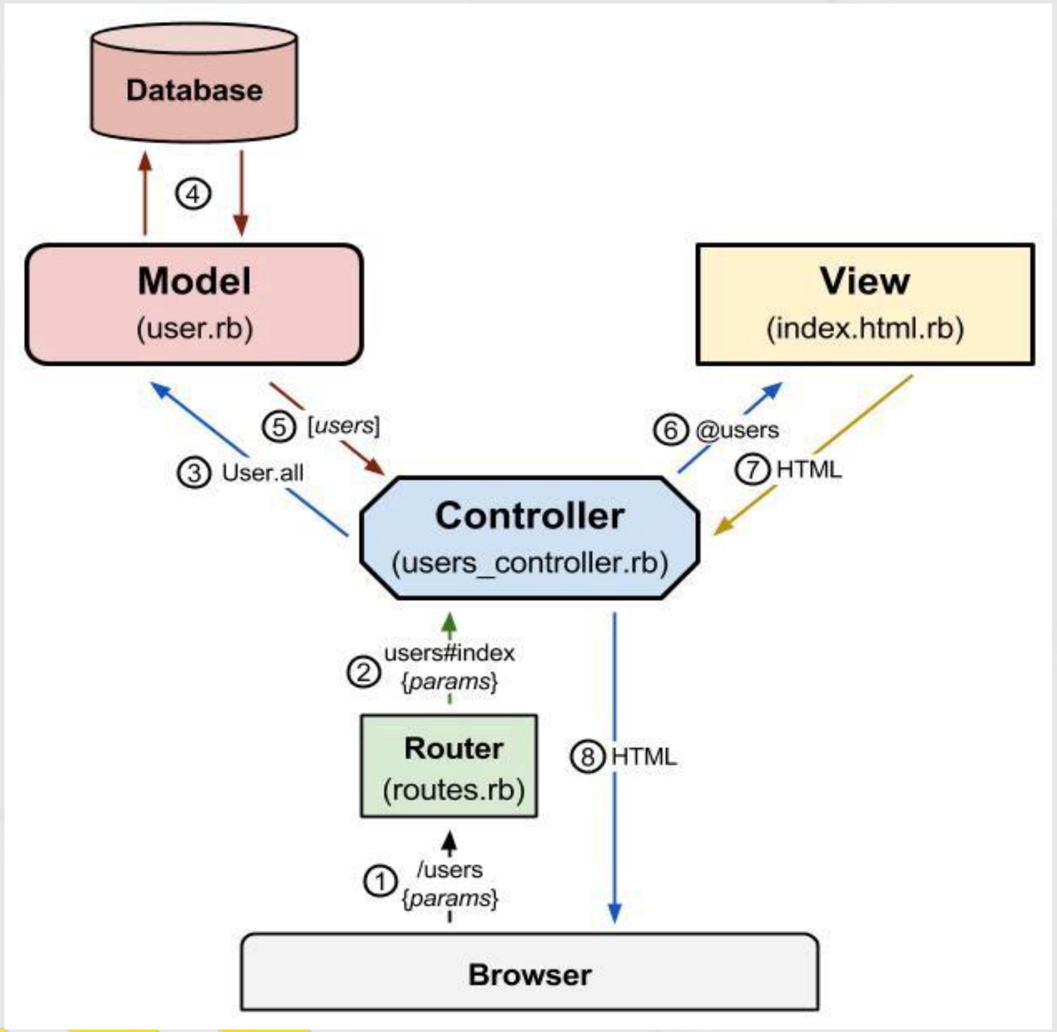
Creating Rails Application in 5 min

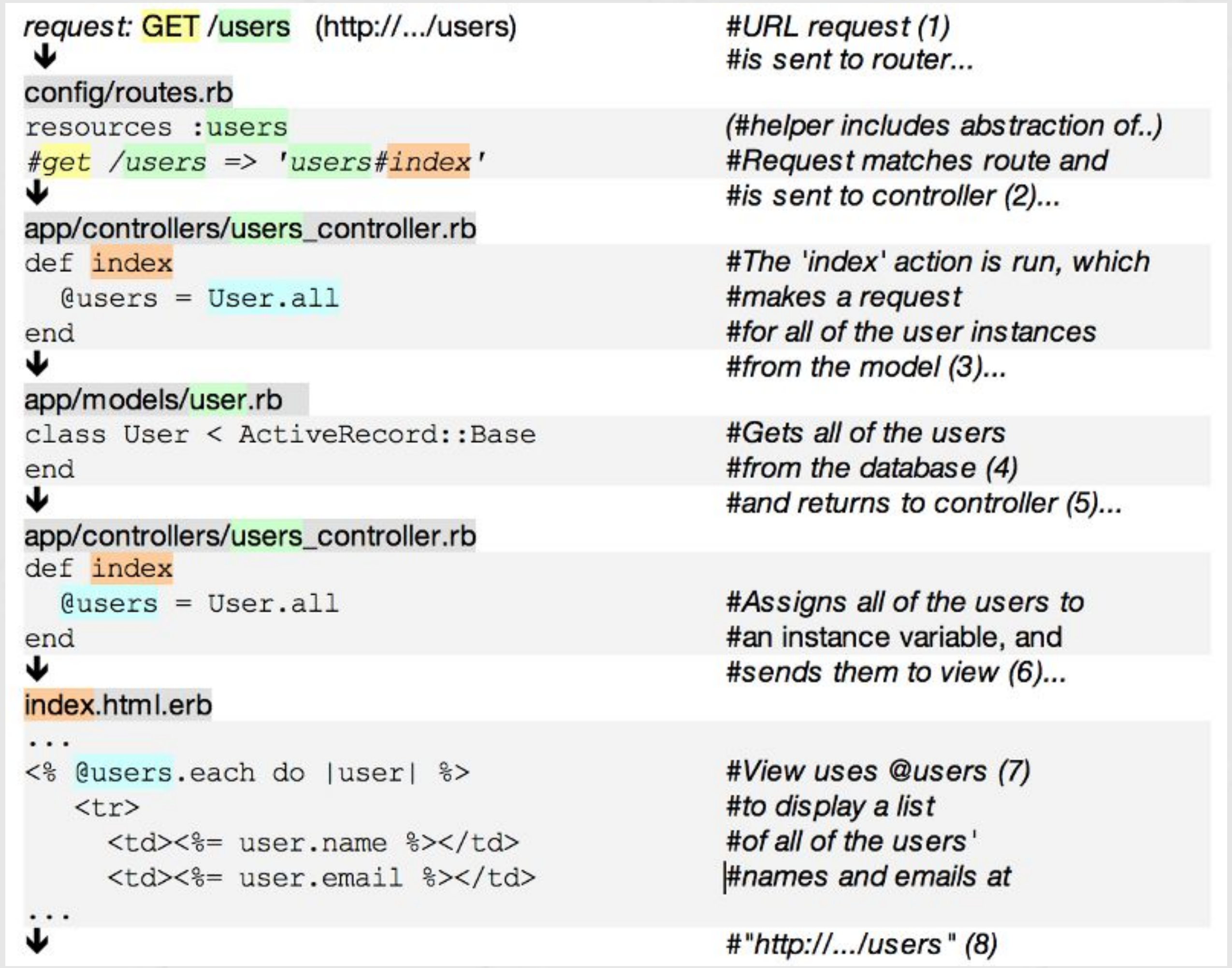
- rails new event_app
- cd event_app
- rails g scaffold events title:string description:text event_date:date
- rake db:create
- rake db:migrate
- rails s

Rails Application folder Structure

- **App** - contain Model, View, Controller, Assets, helper, Mailer
- Bin - contain code to start, setup, update, deploy and run
- **Config** - routes, database and other configurations
- Config.ru - rack configuration to start rack based application
- **Db** - database migration files
- **Gemfile/Gemfile.lock** - gem dependencies required for application
- **Lib** - extended modules
- **Log** - application log
- **Public** - static files
- **Rakefile** - load task that can be run from command line
- **Test** - contain unit testing code
- Tmp - temporary files
- Vendor - third party code

Back to MVC Architecture





```
request: GET /users   (http://.../users)           #URL request (1)
↓                                                   #is sent to router...
config/routes.rb
resources :users                                    (#helper includes abstraction of..)
#get /users => 'users#index'                        #Request matches route and
↓                                                   #is sent to controller (2)...
app/controllers/users_controller.rb
def index                                           #The 'index' action is run, which
  @users = User.all                                 #makes a request
end                                                 #for all of the user instances
↓                                                   #from the model (3)...
app/models/user.rb
class User < ActiveRecord::Base                     #Gets all of the users
end                                                 #from the database (4)
↓                                                   #and returns to controller (5)...
app/controllers/users_controller.rb
def index
  @users = User.all                                 #Assigns all of the users to
end                                                 #an instance variable, and
↓                                                   #sends them to view (6)...
index.html.erb
...
<% @users.each do |user| %>                         #View uses @users (7)
   <tr>                                             #to display a list
     <td><%= user.name %></td>                      #of all of the users'
     <td><%= user.email %></td>                     #names and emails at
...
↓                                                   #"http://.../users" (8)
```

Model

- Business Logic
- Validation
- Before/After action for database
- Associations
- Queries

Controller

- Filtering Parameters of request
- Sending response

SYNERZIP
Agile Software Product Development Partner

View

- Erb template
- Partial View
- Layouts
- Tag_helpers

SYNERZIP
Agile Software Product Development Partner

Routes (HTTP Verbs)

- Get
- Post
- Put
- Patch
- Delete

Routes in Rails

- resources :users

/users - get

/users/:id - getBusiness Logic

/users - post

/users/new - get

/users/:id/edit - get

/users/:id - put/patch

/users/:id - delete

- resource :users

/users/:id - getBusiness Logic

/users - post

/users/new - get

/users/:id/edit - get

/users/:id - put/patch

/users/:id - delete

SYNERZIP
Agile Software Product Development Partner

Routes in Rails

- Get ‘photos/:id’ => “photos#show”
- put ‘photos/:id’ => “photos#update”
- patch ‘photos/:id’ => “photos#update”
- post ‘photos/’ => “photos#create”
- delete ‘photos/:id’ => “photos#destroy”
- match ‘photos/:id’, to:“photos#update”, via: [:put,:patch]
- Root :to => “photos#index”

Creating an CRUD without Scaffold

- rails g model user name:string date_of_birth:date
- Rails g controller users
- rake db:migrate
- rails s

SYNERZIP
Agile Software Product Development Partner

Relationships

- has_many :users
- has_one :user
- belongs_to :user
- has_many_and_belongs_to :users
- has_many :users, :through => :user_events