# Ruby on Rails

Sustainable productivity for web-application development

# Rails

1. Philosophy
2. History
3. MVC
4. What’s Inside

# Philosophy of Rails

DRY

CoC (Convention over configuration)

SRP (Single responsibility principle)

Decouple

Opinionated

Community

# History of Rails

Language 1995

Framework 2004

# History

| | |
|---|---|
| Rails 0 | 07 / 2004 |
| Rails 1.0 | 12 / 2005 |
| Merb | 12 / 2008 |
| Rails 3.0 | 08 / 2010 |
| Rails 4.0 | 06 / 2013 |
| Rails 5.x | Fall 2015 |

```
MVC.describe!
```

# MVC

- Architectural Pattern
- 1979
- Adapted for Web
- Popularized for web by STRUTS

# Okay, but WHY?

## Why do we need a pattern?

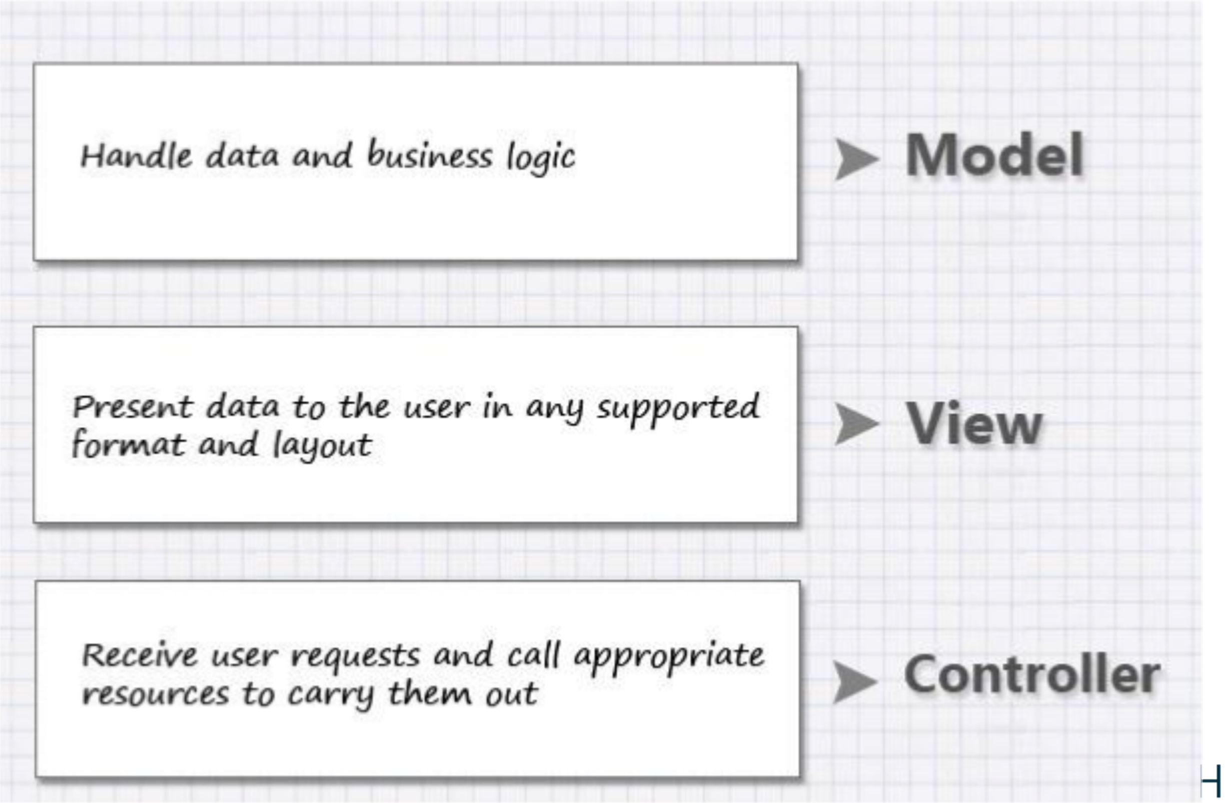
- HTTP
- HTML
- CSS
- JavaScript
- Database
- Tests
- Support Libraries
- Configuration
- Caching
- And more ...

# To avoid “spaghetti code”

# Back to MVC

Microsoft ASP.net MVC3
GRAILS
ZF ZEND FRAMEWORK
CakePHP
CF WHEELS
django
Struts
spring
LARAVEL
RAILS
symfony
sails
Flask
web development, one drop at a time
CodeIgniter
Express

# mvc: [:model, :view, :controller]

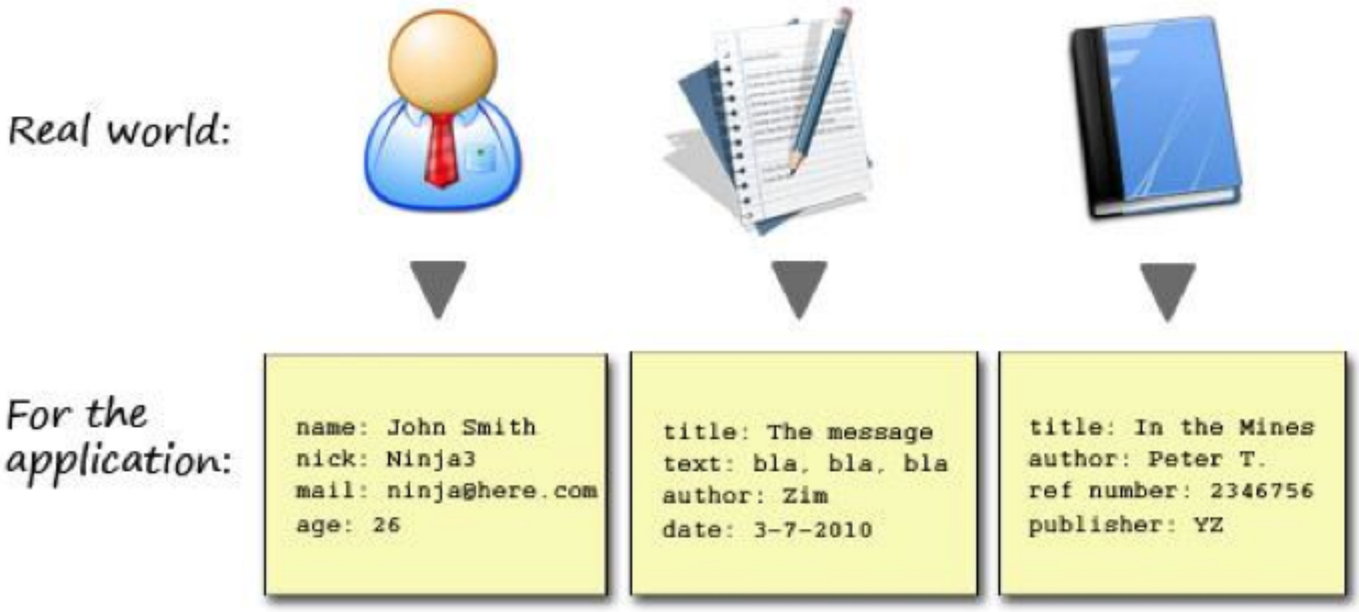


Handle data and business logic ➤ Model

Present data to the user in any supported format and layout ➤ View

Receive user requests and call appropriate resources to carry them out ➤ Controller

# Model



Real world:

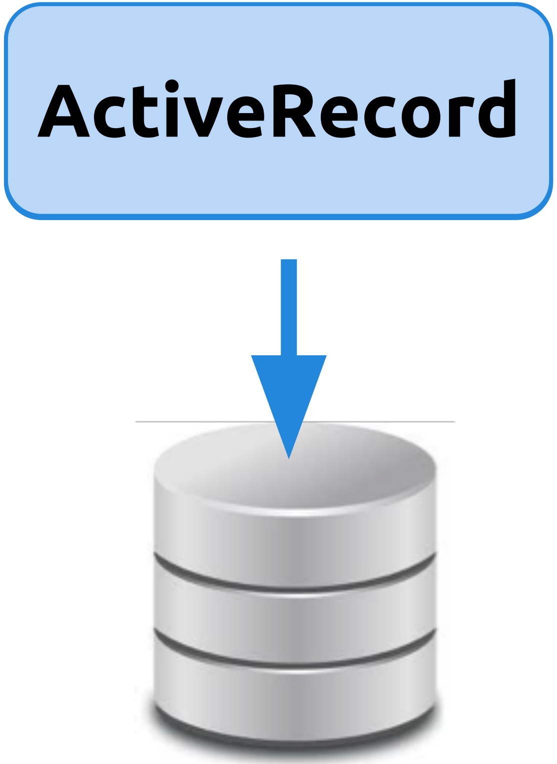
For the application:

name: John Smith
nick: Ninja3
mail: ninja@here.com
age: 26

title: The message
text: bla, bla, bla
author: Zim
date: 3-7-2010

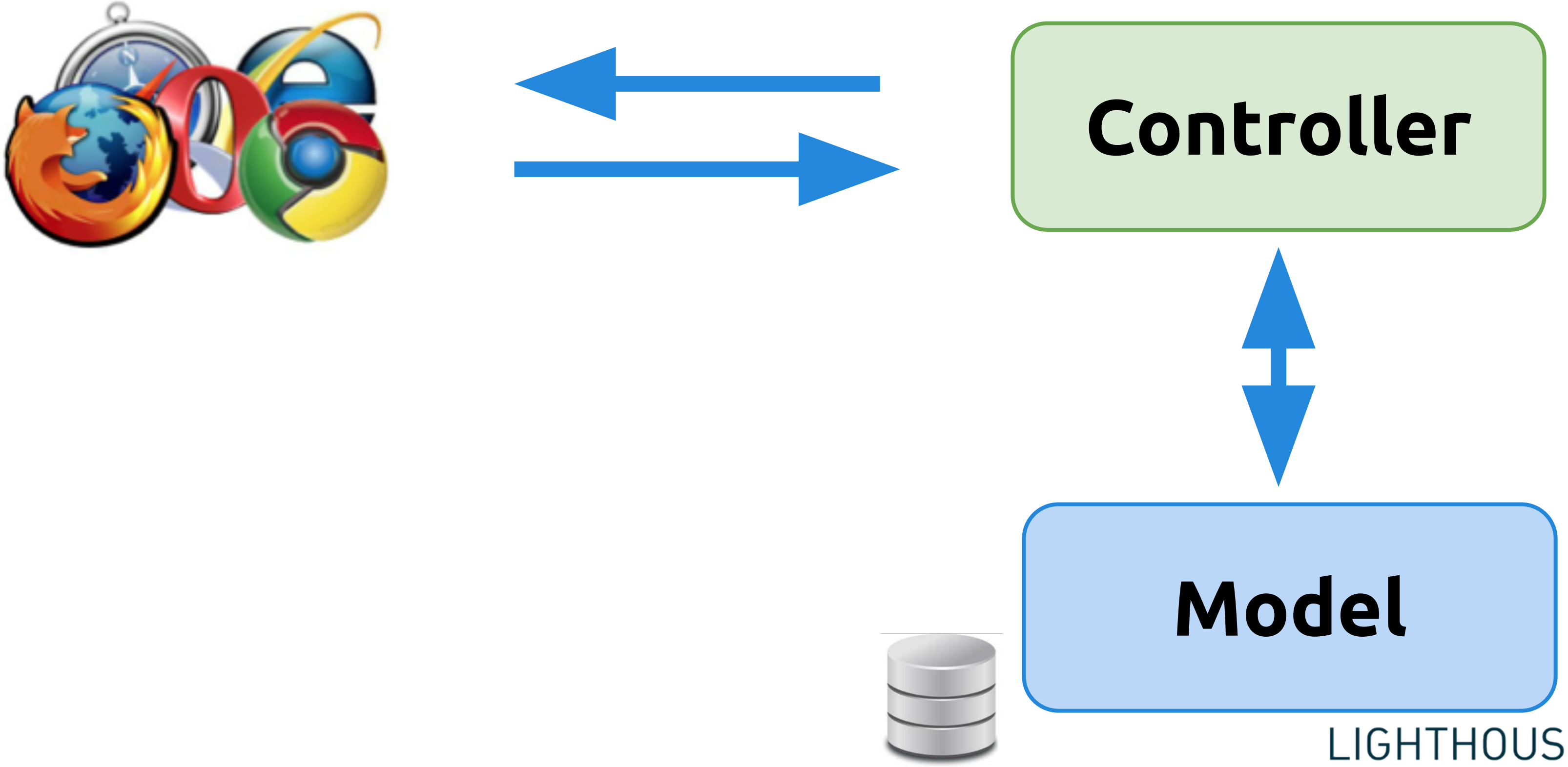
title: In the Mines
author: Peter T.
ref number: 2346756
publisher: YZ

# Model



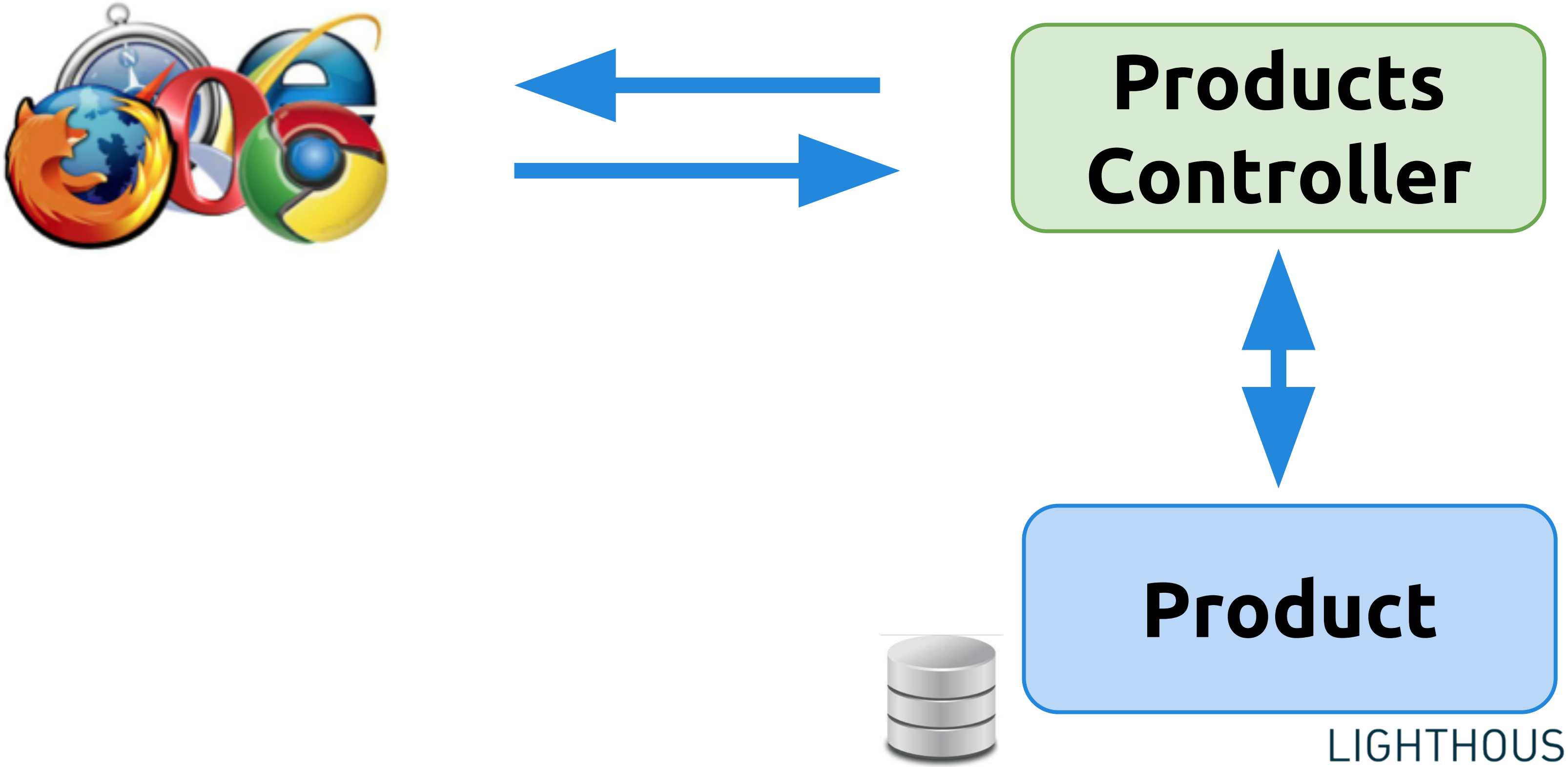
ActiveRecord

# Controller



LIGHTHOUSE LABS

# Controller



# Controller Actions

Each **Controller** manages one model (noun)

Each **Action** does something with that model (method, verb)

Multiple actions can occur on a model (eg: Product):

# View

HTML 5

JSON

JavaScript Object Notation

CSV

<XML>

PDF

Haml

jade

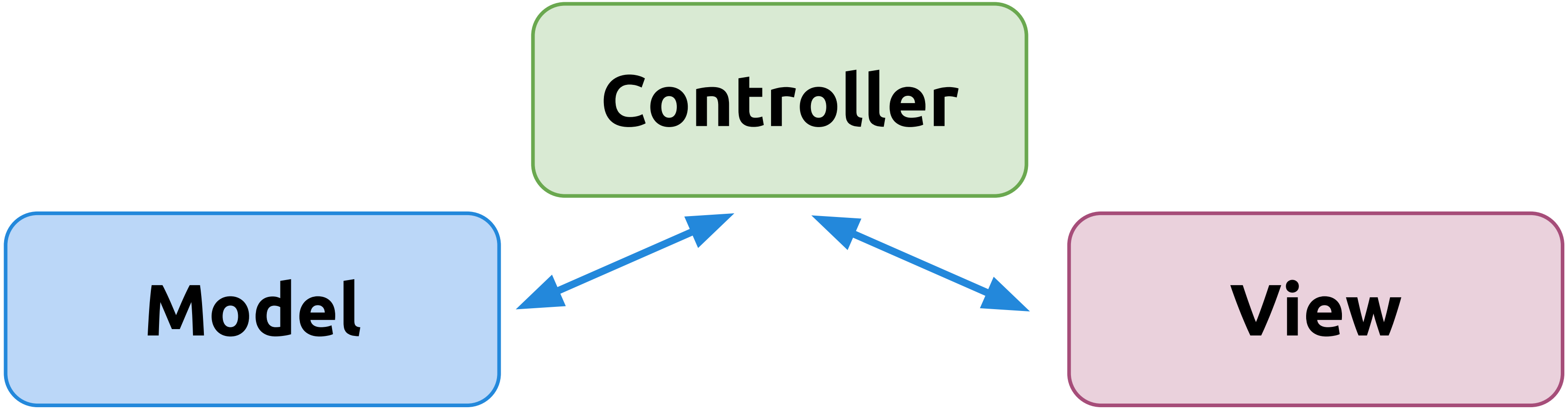
Node Template Engine

# Still don’t understand Controller!

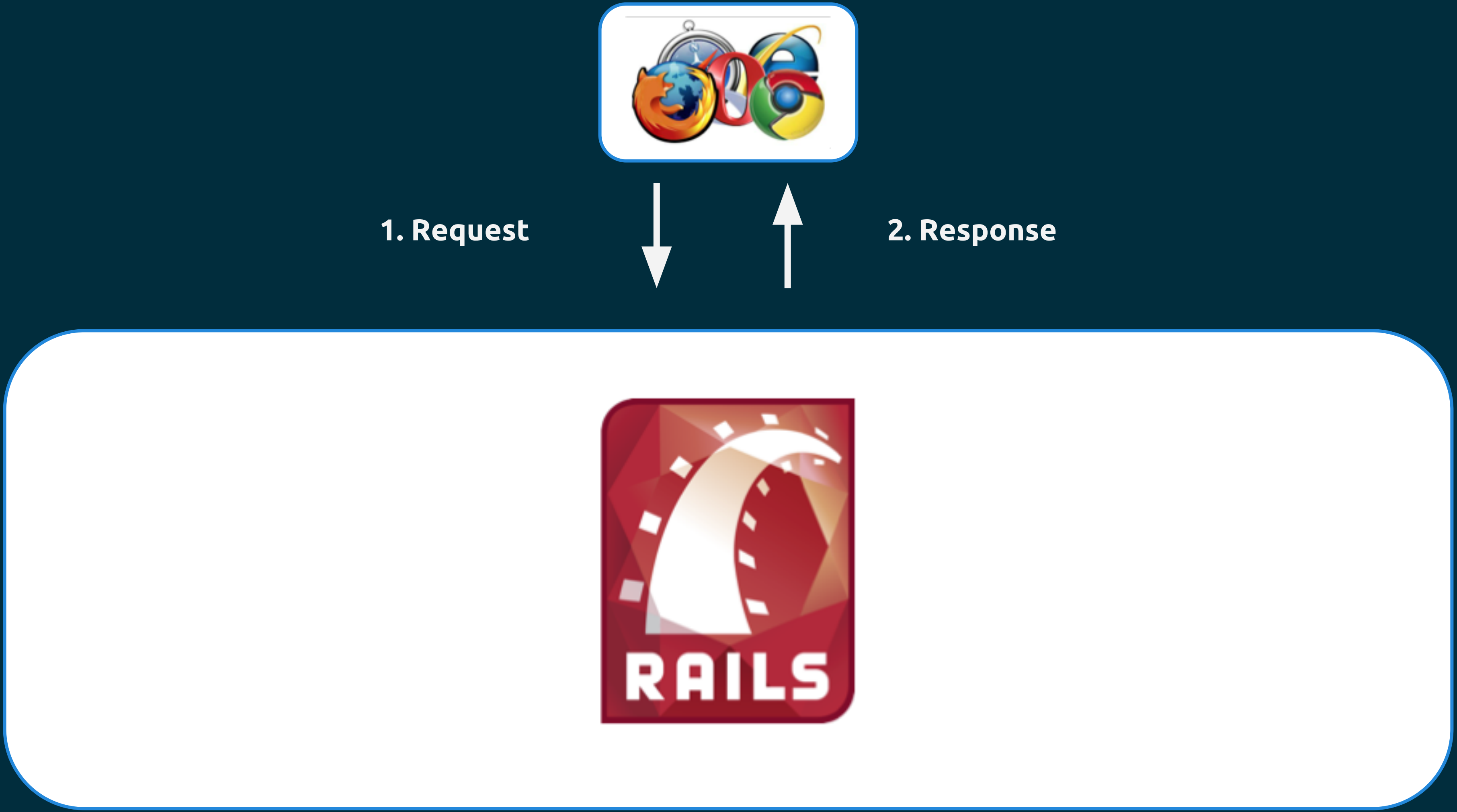
Handles the user **Request** then **Responds** to the user

Uses the **Model** to handle the Request

Uses the **View** to Respond

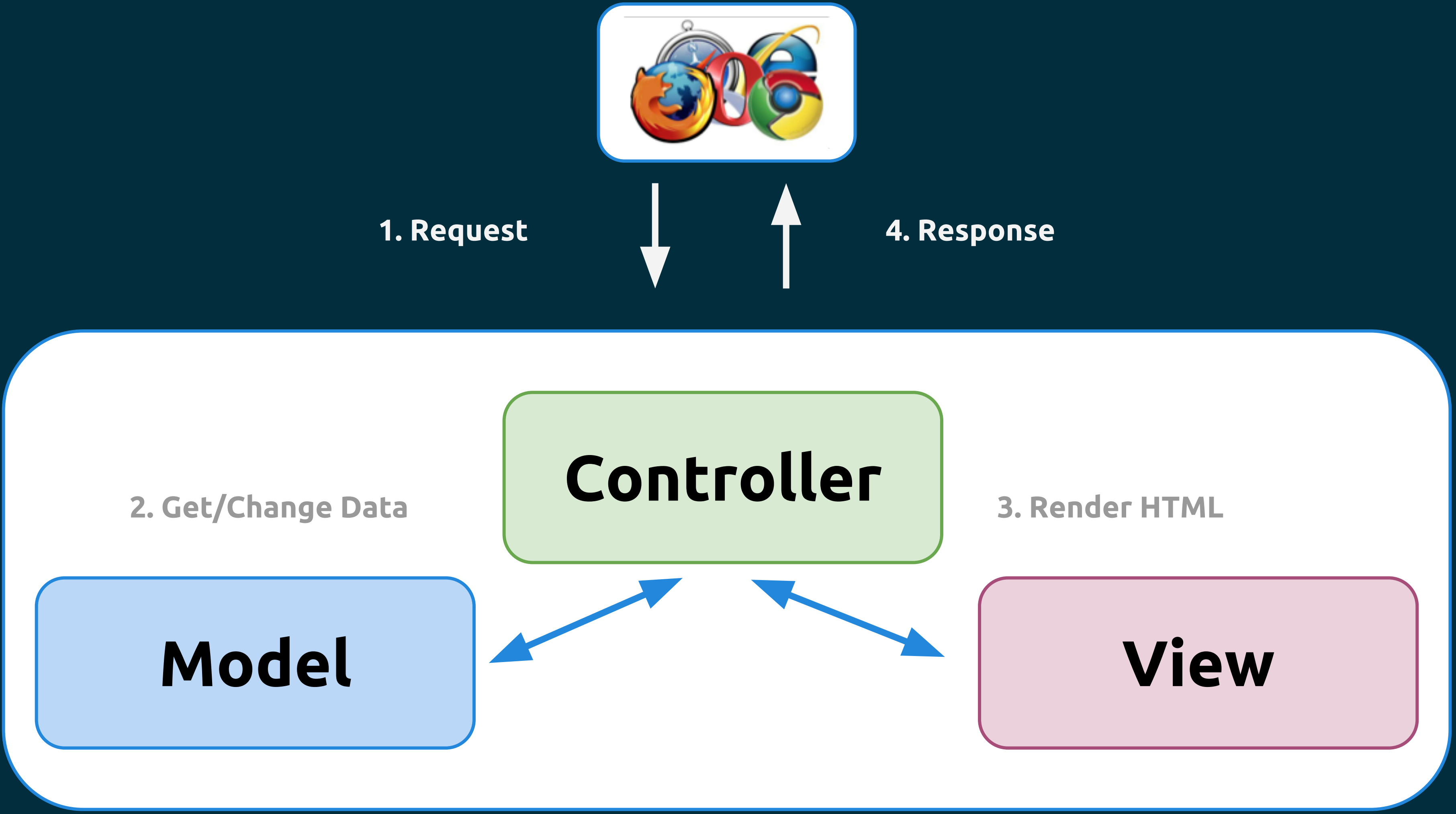


# All Together Now



1. Request

2. Response



1. Request

4. Response

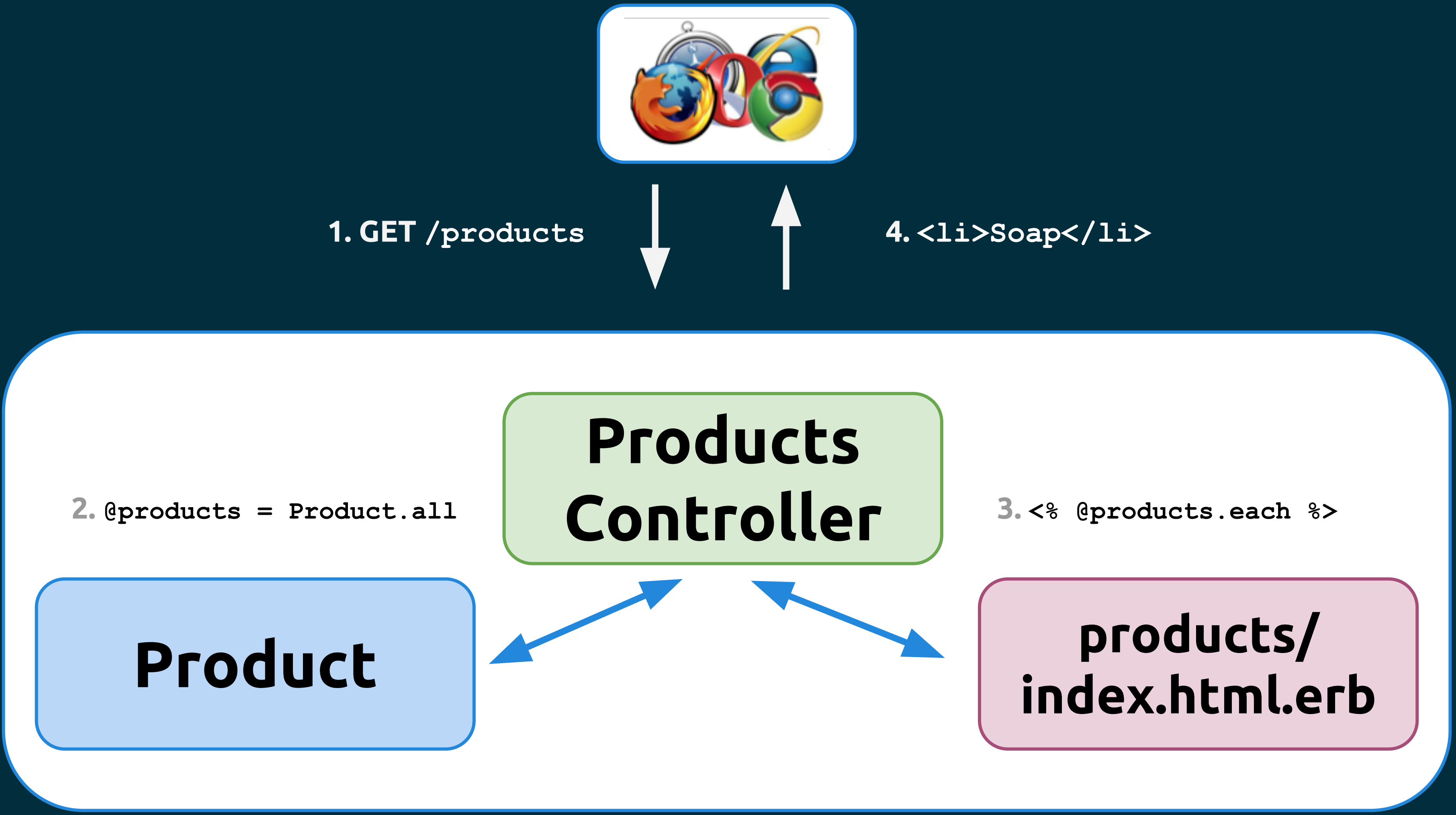
Controller

2. Get/Change Data

3. Render HTML

Model

View



1. GET /products

4. <li>Soap</li>

2. @products = Product.all

Products Controller

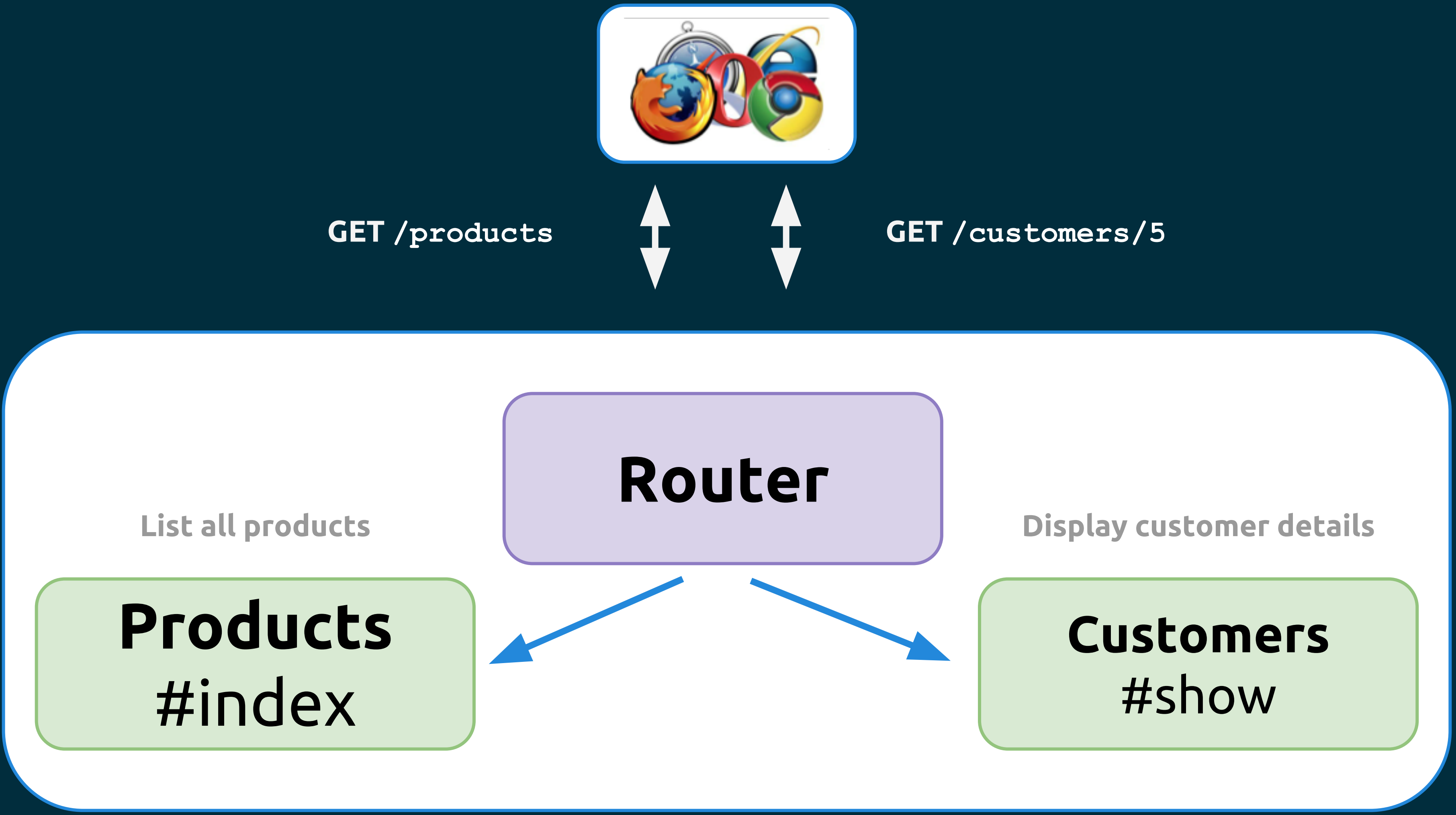
3. <% @products.each %>

Product

products/index.html.erb

# The Router

# Which controller#action does the User want to access?



GET /products

GET /customers/5

Router

List all products

Products
#index

Display customer details

Customers
#show

# What’s Inside

| | |
|---|---|
| **ActiveRecord** | Associations, Validations, Persistence, Migrations |
| **ActionController** | Actions, Filters, Params, Render, Redirect, etc... |
| **ActionView** | Templates, Rendering (ERB), Helpers |
| **ActionDispatch** | Routing, HTTP, Request, Response, Cookies, Sessions |
| **ActiveSupport** | Utility Classes, Extends Ruby (makes it even nicer) |
| **ActionMailer** | Sending Email |

:)