intro to

Ruby

programming language

Rails

a full-stack

framework for building web apps

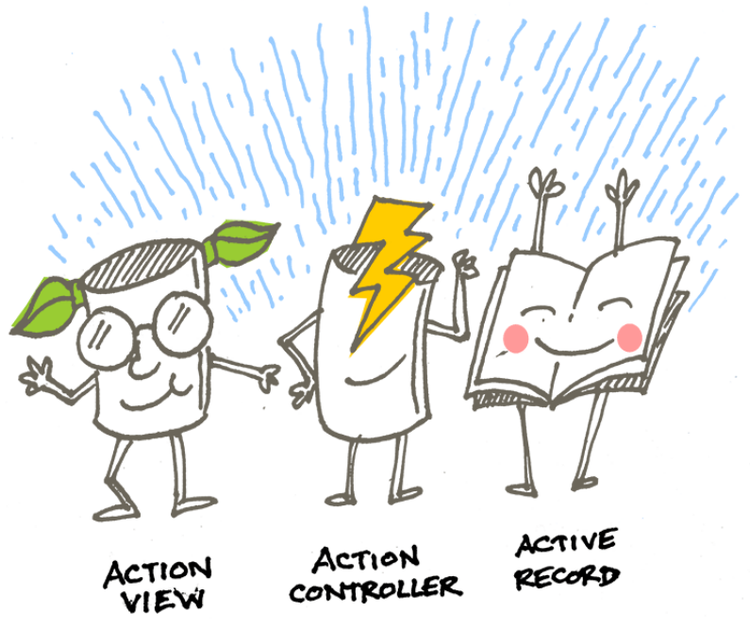
FONDAZIONE
BRUNO KESSLER

ict 4g

Rails is

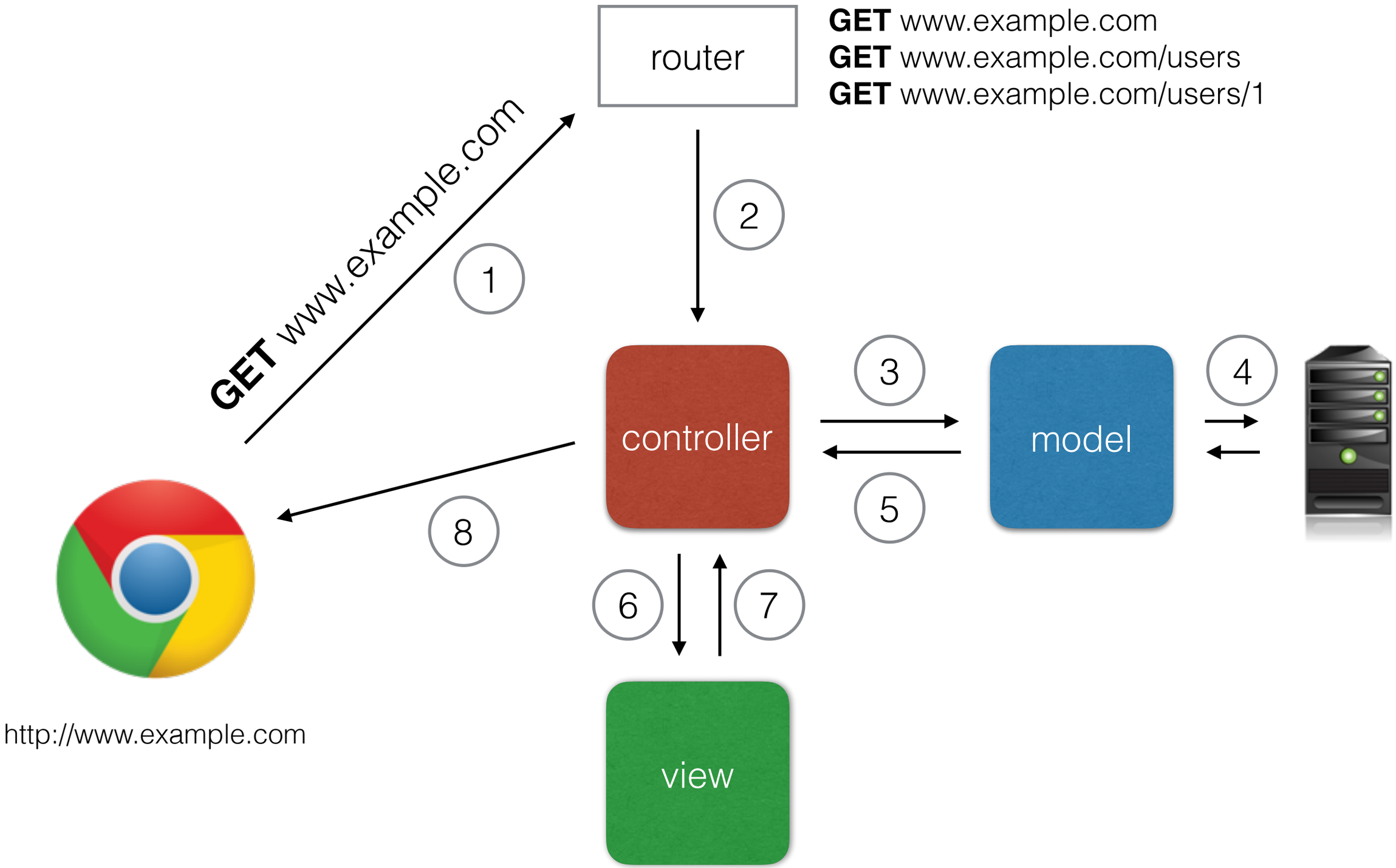
- Open Source (released in 2004)
- (Very) Opinionated
- Used by hundreds of thousands of web applications like Basecamp, GitHub, Shopify, Airbnb, Twitch, SoundClou, Hulu, Zendesk, Square, Highrise, Cookpad, …

Rails gives you everything you need



- Abstraction of the data layer (model)
- Management of the request/response flow (controller)
- Data display in HTML, JSON, XML, … (view)
- Asset management (images, stylesheets, Javascript)

Rails MVC



GET www.example.com
GET www.example.com/users
GET www.example.com/users/1

Convention over Configuration

Who cares what format your database primary keys are described by?

Does it really matter whether it's `id`, `postId`, `posts_id`, or `pid`?

Is this a decision that’s worthy of recurrent deliberation? **No.**

Example

if we have an entity called Book, Rails will assume the following directory structure

```
app
|-- controllers
|   `-- books_controller.rb
|-- models
|   `-- book.rb
`-- views
    `-- books
        |-- index.html.erb
        |-- new.html.erb
        |-- edit.html.erb
        `-- ...
```

FONDAZIONE BRUNO KESSLER

ict 4g

Example/2

and will assume the following table structure in the DB

Example/2

and will assume the following table structure in the DB

books
id
created_at
updated_at
title
…

Example/2

and will assume the following table structure in the DB

books
id
created_at
updated_at
title
…

author
id
created_at
updated_at
name
…

FONDAZIONE BRUNO KESSLER

ict 4g

Example/2

and will assume the following table structure in the DB

books
id
created_at
updated_at
title
…
author_id

author
id
created_at
updated_at
name
…

FONDAZIONE BRUNO KESSLER

ict 4g

DB Abstraction

- Rails uses ActiveRecord/Mongoid to abstract the DB engine
- Default DB engine: SQLite
- The day you want to switch to another engine you change one line of code*
- You don’t need to write SQL (but you need to know SQL)

ActiveRecord Example

```
Book.all
# SELECT  "books".* FROM "books" LIMIT ?  [["LIMIT", 11]]

Book.create title: 'The Rainbow Nation', publisher: 'Einaudi'
# INSERT INTO "books" ("title", "publisher", "created_at", "updated_at")
# VALUES (?, ?, ?, ?)  [["title", "The Rainbow Nation"], ["publisher",
# "Einaudi"], ["created_at", "2017-10-03 15:18:53.858906"], ["updated_at",
# "2017-10-03 15:18:53.858906"]]
```

DB Migrations

- Allow for incremental development
- Simplify collaboration
- Enables rollback in case of errors
- Simpler deployments

Example

Create a new entity called Book

```
class CreateBooks < ActiveRecord::Migration
  def change
    create_table :books do |t|
      t.string :title
      t.string :publisher
      t.timestamps
    end
  end
end
```

Later add a reference to the author

```
class AddAuthorToBooks < ActiveRecord::Migration
  def change
    add_reference :books, :author, foreign_key: true
  end
end
```

Environments

- **development** on your machine
- **test** to run automated tests
- **production** on your server

with their own DB!

Testing Environment

- Minitest - comes by default with any Rails app = no configuration needed
- RSpec - standard *de-facto*
- Cucumber - human readable test specifications (based on RSpec)

Gems

- You don’t want to re-invent anything
- Almost everything has been done before
- Examples: authorization, authentication, emails, attachments, forms, PDF generation, geolocation, localization, image manipulation, ….

Generators

- Command line scripts for the developer to generate the skeleton of common components

- Want to see it in action? Let’s build a web application in 3 minutes