🔹 Rotas (Routes)

As rotas no Rails são definidas no arquivo `config/routes.rb` e determinam como as requisições HTTP (GET, POST, PUT, DELETE, etc.) serão mapeadas para os controllers e suas respectivas actions. A definição de rotas permite associar um caminho de URL a um comportamento no sistema, e o Rails é altamente flexível nesse aspecto.

📌 Explicação Técnica Detalhada

```ruby
Rails.application.routes.draw do

  get '/', controller: 'home', action: 'index'

end
```

1. **`Rails.application.routes.draw`**:
   Esse é o método central que configura as rotas da aplicação. Dentro dele, as rotas são declaradas. O Rails usa **DSL (Domain Specific Language)** para configurar rotas de forma concisa e expressiva.
2. **`get '/'`**:
   Aqui estamos definindo uma rota do tipo **GET** para o caminho raiz (`/`). O método `get` indica que essa rota irá lidar com requisições GET, que são típicas para a exibição de conteúdo (como páginas HTML). O `/` representa a URL do site, ou seja, a página inicial.
3. **`controller: 'home', action: 'index'`**:
   Aqui estamos associando a requisição GET para `/` à **action** `index` no **controller** `HomeController`.
   - **Controller**: O nome do controller é **home**, que corresponde ao arquivo `home_controller.rb` dentro de `app/controllers`. Em Rails, os nomes de controller seguem a convenção de serem no plural, mas no caso da página inicial, usaremos o nome `HomeController`.
   - **Action**: O nome da ação é **index**, que é um método dentro do controller que será executado quando a requisição for recebida.

Assim, toda requisição GET para a URL `/` será mapeada para o **HomeController** e sua **action `index`**. O Rails segue a convenção de associar URLs com métodos que executam ações específicas, facilitando o roteamento de maneira eficiente.

🔹 Controller

O **controller** é o ponto central que lida com a lógica de negócios e a manipulação dos dados antes de enviá-los à view. Ele lida com a requisição HTTP, processa dados e direciona para a view, podendo também interagir com o **model** para buscar ou manipular dados do banco de dados.

📌 Explicação Técnica Detalhada

```ruby
class HomeController < ApplicationController
  def index
    @nome = "USER MASTER"
    # @nomes = ["Alice", "Bob", "Carlos", "Diana"]

    @devs = [
      { nome: 'Weslley',  dev: 'Ruby on Rails' },
      { nome: 'André',    dev: 'Next.js' },
      { nome: 'Kaike',    dev:'Mysql'}
    ]
  end
end
```

1. **Herança de `ApplicationController`**:
   O **HomeController** herda de **ApplicationController**, que é a classe base para todos os controllers em Rails. Isso permite que o `HomeController` herde funcionalidades globais, como autenticação, filtros e métodos auxiliares definidos em `ApplicationController`. Além disso, **ApplicationController** centraliza configurações e comportamentos compartilhados entre os controllers.

2. **Método `index`**:
   Este é o método que será chamado quando a rota GET para `/` for acessada. Cada método de um controller é uma **action** que pode processar uma requisição. Nesse caso, o método `index` é responsável por processar a lógica para a página inicial.

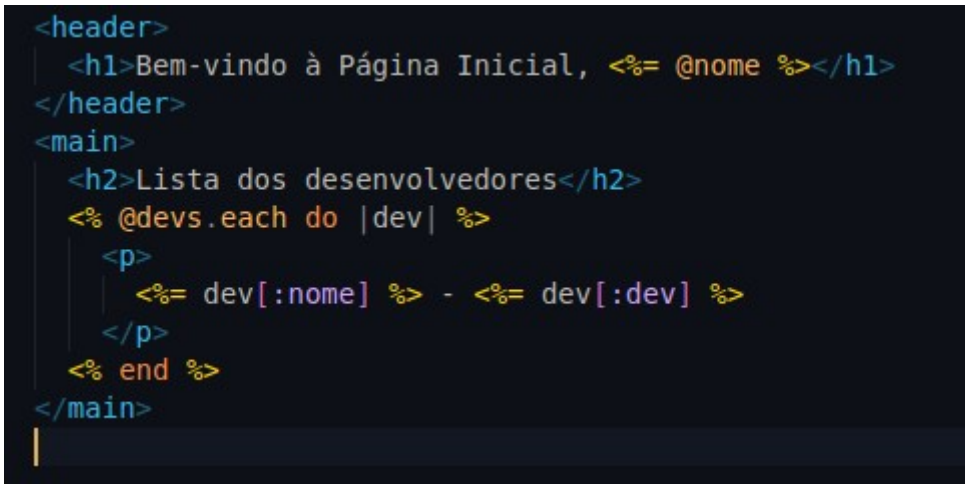
3. **Variáveis de Instância**:
   - **`@nome = "USER MASTER"`**:
     Esta variável de instância (`@nome`) é definida no controller e estará disponível para a view correspondente. Variáveis de instância são prefixadas com `@` e são passadas automaticamente do controller para a view. O valor de `@nome` será acessado na view para renderizar o texto dinâmico.
   - **`@devs`**:
     A variável `@devs` contém um array de hashes, onde cada hash representa um desenvolvedor. Cada desenvolvedor tem um `:nome` e um `:dev`, que são usados para descrever o nome do desenvolvedor e sua especialidade técnica. Esta variável de instância também é passada para a view, onde pode ser usada para gerar uma lista dinâmica.

4. **Interação com o Model**: Embora não esteja interagindo diretamente com um model no exemplo acima, o controller pode acessar os models (que representam dados do banco de dados) e passá-los para a view. Em um cenário mais complexo, o controller poderia buscar dados de um banco de dados usando os **models** e passá-los para a view.

🔹 View (ERB - Embedded Ruby)

A **view** é responsável por renderizar o HTML final que será retornado ao navegador. No Rails, as views são arquivos `.html.erb`, que misturam HTML com código Ruby. O código Ruby pode ser inserido na view através das tags **ERB**.

📌 Explicação Técnica Detalhada



1. **Tags ERB**:
   - **`<%= %>`**: A sintaxe `<%= %>` é usada para **exibir o resultado de uma expressão Ruby no HTML**. Dentro dessas tags, o Ruby é executado e o resultado é convertido em texto e inserido no HTML. Por exemplo, `<%= @nome %>` renderiza o valor de `@nome` na página.
   - **`<% %>`**: A sintaxe `<% %>` é usada para **executar código Ruby sem exibi-lo diretamente**. Esse tipo de tag é utilizado quando você deseja fazer iterações, condições ou executar outras operações sem gerar saída no HTML.
2. **Exibição de Variáveis de Instância**:
   - **`<%= @nome %>`**: Dentro da tag `<h1>`, o valor da variável `@nome` é exibido. Quando o método `index` é chamado no controller, `@nome` é definida como `"USER MASTER"`. Portanto, a view renderiza a string `"USER MASTER"` dentro da tag `<h1>`.
   - **`<%= dev[:nome] %>` e `<%= dev[:dev] %>`**: Dentro do loop, os valores de cada hash dentro de `@devs` são acessados. O loop itera sobre o array `@devs`, e para cada `dev`, ele exibe o nome do desenvolvedor e a sua especialidade. O hash é acessado com as chaves `:nome` e `:dev`.
3. **Looping com `each`**:
   - **`<% @devs.each do |dev| %>`**: O método `each` é um iterador que percorre o array `@devs`. Para cada iteração, a variável `dev` armazena o hash de um desenvolvedor. O loop continua até que todos os itens no array sejam processados.

4. **Estrutura HTML**: A estrutura HTML foi definida de forma simples. Dentro da tag `<header>`, temos o título com o nome do usuário, e no `<main>`, exibimos uma lista de desenvolvedores e suas especialidades.

🔹 Fluxo Completo: Da Requisição à Resposta

1. **Requisição do Usuário**: O usuário acessa `http://localhost:3000/` no navegador. O navegador faz uma requisição HTTP **GET** para essa URL.
2. **Roteamento (Routing)**: O Rails verifica as rotas definidas em `config/routes.rb` e encontra que a URL `/` deve ser direcionada para a **action `index`** do **controller `HomeController`**.
3. **Controller**: O Rails executa a **action `index`** no `HomeController`. Dentro dessa action:
    - O Rails prepara os dados necessários para a view, definindo as variáveis de instância `@nome` e `@devs`.
    - Esses dados são passados para a view associada à action `index`.
4. **View**: O Rails renderiza a view `index.html.erb`, substituindo as variáveis de instância `@nome` e `@devs` pelos seus valores reais. O HTML é gerado com as informações dinâmicas.
5. **Resposta ao Usuário**: O HTML gerado pela view é enviado de volta para o navegador do usuário, que exibe a página inicial com o nome do usuário e a lista de desenvolvedores.

Resumo Final:

- **Rota**: Mapeia URLs para controllers e actions específicas.
- **Controller**: Processa a requisição, interage com o modelo (se necessário), e prepara os dados para a view.
- **View**: Exibe os dados processados pelo controller, gerando HTML dinâmico através do ERB.