**一、提供中间件功能**

- call(env) 实现路由的识别与分发

通过Journey实现路由的识别与分发

**二、提供路由的集合**

路由集内的路由被用于识别与分发

**1路由集信息**

**路由集**

routes 路由的集合(无名路由、具名路由)

路由集由Journey::Routes实现

named\_routes 具名路路由集，routes的子集

**2路由集管理**

**绘制路由表的默认参数**

resources\_path\_names 默认action名对应URL出现的名称

default\_scope

**路由表绘制(添加多个路由)**

disable\_clear\_and\_finalize 绘制禁用clear!和finalize!，默认false

- draw(&block) 绘制路由表

清理路由表

clear! unless disable\_clear\_and\_finalize

绘制路由表

eval\_block(block)

标记结束绘制

finalize! unless disable\_clear\_and\_finalize

- eval\_block(block) (私有)具体实现路由表的绘制功能

在Mapper实例域中执行block，实现路由表block的绘制。见Mapper

- clear! 清理路由集中所有路由

该方法还绘制由prepend方法添加的前置路由表(默认前置路由表为空)

- finalize! 标记绘制路由表结束

该方法还绘制由append方法添加的后置路由表(默认后置路由表为空)

**添加一个路由**

- add\_route(app, \*args) 添加路由

**3路由集正解析—路由识别(根据path、method生成参数)**

- recognize\_path(path, environment = {}) 返回解析出的参数

包括controller、action等

**4路由集反解析(根据路由生成URL)**

**路由url\_for反解析默认参数**

default\_url\_options 用url\_for方法生成URL的默认参数

默认为空哈希{}

**路由反解析方法**

- path\_for(options, route\_name = nil) 根据参数匹配的路由生成path

- url\_for(options, route\_name = nil) 根据参数匹配的路由生成URL

**5路由集反解析方法模块**

- url\_helpers 当前路由集的反解析方法模块

模块extend自身，因此可通过模块方法调用如下反解析方法

·具名路由反解析方法

·路由集url\_for反解析方法

例：routes.url\_helpers.articles\_path # => "/articles" (具名路由对应的path)

例：class Xiao; include Rails.application.routes.url\_helpers; end

Xiao.new.articles\_path # => "/articles" (具名路由对应的path)

例：routes.url\_for(...) # => 返回反解析的URL

**挂载反解析方法模块**

- define\_mounted\_helper(name) 挂载当前路由集的反解析模块

将当前路由集反解析模块挂载到MountedHelpers的name方法

- mounted\_helpers 返回MountedHelpers模块

某类扩展该模块后，调用挂载方法时将返回对应反解析模块。

MountedHelpers#main\_app挂载了Rails应用对应的路由反解析模块