**一、使用**

**1 config/initializers/omniauth.rb实现**

# 添加全局配置见Omniauth

OmniAuth.config.before\_request\_phase do

# do something ...

end

# 向Rails中加入中间件

Rails.application.config.middleware.use OmniAuth::Builder do

provider :ciprun, "123050457758183", "VR/+DtqFA6hTtk2sueilzw=="

end

**2验证通过后获取用户信息**

env['omniauth.auth'] 返回认证信息，OmniAuth::AuthHash对象

**二、全局配置**

**1全局配置对象(Omniauth)**

Omniauth.config Omniauth::Configuration单例对象

**2全局配置方法(Omniauth::Configuration)**

- add\_camelization(name, camelization) 为Builder::provider提供对应类名

path\_prefix 请求第三方认证路径前缀，默认/auth

/auth/:provider将使用provider对应的中间件实现第三方认证

- on\_failure(&block) 认证报错时调用的中间件

默认OmniAuth::FailureEndpoint

- before\_request\_phase(&block) 认证请求的前置过滤器

request\_phase前执行block.call(env)

- before\_callback\_phase(&block) 认证回调的前置过滤器

callback\_phase前执行block.call(env)

**三、中间件封装器Omniauth::Builder**

可将多个第三方认证Rack封装到统一的Rack中，继承自Rack::Builder

- initialize 同Rack::Builder

- provider(klass, \*args, &block) 类似use(:klass, \*args, &block)

klass Omniauth.config#add\_camelization注册的中间件

args(非hash部分) 提供中间件的配置，与Provider.args组成配置

args(hash参数) 提供中间件的配置

:setup 认证(请求、回调)阶段前被执行的block，默认false

:path\_prefix 识别认证请求、回调的路径前缀

:request\_path 识别认证请求的路径

:callback\_path 识别认证回调的路径

... 其他配置

block{|optiosn| } block被执行在klass实例作用于中

Rails.application.config.middleware.use OmniAuth::Builder do

provider :ciprun, "123050457758183", "VR/+DtqFA6hTtk2sueilzw=="

provider :weibo, "ssss", "ssssss"，setup: lambda{} do |options|

options[:path\_prefix] = '/autherize' # options为中间件配置

end

end

**四、认证中间件—Provider (例OmniAuth::Strategies::Ciprun)**

混入OmniAuth::Strategy模块的类，即成为一个认证中间件

**1中间件配置信息**

**中间件全局配置**

+ default\_options 返回中间件全局配置信息

由类实例变量保存的Hash簇对象

+ option(name, value=nil) 设置default\_options全局配置信息

指定的配置可被provider的hash参数覆盖

**中间件的配置**

+ args(keys=nil) 指定keys与初始化参数组成中间件配置

配置被存入options中

options 中间件配置属性

初始化中被设置，并有如下优先级

·Omniauth::Builder#provider中非hash参数提供的配置

·Omniauth::Builder#provider中hash参数提供的配置

·Provider全局配置(由Provider.option方法设置的)

**中间件属性**

app 保存中间件的底层app

env 保存调用的环境变量

**2中间件初始化**

- initialize(app, \*args, &block) 初始化一个Provider中间件

参数见Omniauth::Builder#provider参数

**3 Rack接口实现**

**Rack接口方法**

- call(env) Rack接口

·@env['omniauth.strategy'] = self

·识别认证请求路径，若满足返回认证请求阶段

·识别认证回调路径，若满足返回认证回调阶段

·若都不满足，则进入底层中间件流程

**认证请求**

- on\_request\_path? 识别认证请求

·判断path匹配options[:request\_path] 或

·匹配"/#{options[:path\_prefix] || OmniAuth.config.path\_prefix}/#{name}"

·若满足则调用认证请求并返回

- request\_call 认证请求

·执行options[:setup].call(env)

·执行过滤器OmniAuth.config.before\_request\_phase.call(env)

·执行request\_phase provider需实现的部分

**认证回调**

- on\_callback\_path? 识别认证回调

·判断path匹配options[:callback\_path]或

·匹配"/#{options[:path\_prefix] || OmniAuth.config.path\_prefix}/#{name}/callback"

·若满足则调用认证回调并返回

- callback\_call 认证回调

·执行options[:setup].call(env)

·执行过滤器OmniAuth.config.before\_callback\_phase.call(@env)

·执行callback\_phase

- callback\_phase 回调处理，provider需重构

·设置env['omniauth.auth']为OmniAuth::AuthHash对象

·进入底层中间件

**3指定认证后数据获取逻辑**

+ uid(&block) 指定获取用户id的block

block被执行的作用于为中间件实例

+ info(&block) 指定获取用户信息的block

+ extra(&block) 指定获取额外信息的block

+ credentials(&block) 指定获取授权信息的block