**文档名称**

**摘要:** 描述文档摘要。

**知识产权申明**

此文档包含上海移宇科技有限公司知识产权信息。

未经书面授权，不得披露，复制，使用此文档的任何数据。

文档编码 :

文档版本号 :

电子文档名称 :

作者 :

创建时间 :

修改人:

最后修改日期:

**历史修改记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 编撰及修订者 | 修订说明 |
| 2013-1-8 | V0.1 | Jing Li | 创建文档 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1.【章节名称】 4](#_Toc369338149)

[1.1 4](#_Toc369338150)

[1.2 4](#_Toc369338151)

[1.3 4](#_Toc369338152)

[1.4 4](#_Toc369338153)

[2. 【章节名称】 5](#_Toc369338154)

[2.1. 5](#_Toc369338155)

[2.2 5](#_Toc369338156)

[3. 【章节名称】 6](#_Toc369338157)

[4. 【章节名称】 7](#_Toc369338158)

[5. 【章节名称】 8](#_Toc369338159)

[6.【章节名称】 9](#_Toc369338160)

[附录1 10](#_Toc369338161)

[附录2 11](#_Toc369338162)

# 1.【章节名称】

## 1.1

## 1.2

## 1.3

## 1.4

# 2. 【章节名称】

## 2.1

## 2.2

# 3. HTTP

## 3.1 Cookie



服务器的响应可以有多个Set-Cookie字段，其中每个字段描述一个Cookie。

# 4. 登录管理flask\_login

## 4.1 current\_user

current\_user是全局变量，用来表示当前有没有登录的用户。

这个全局变量是flask\_login插件的核心。

current\_user = LocalProxy(**lambda**: \_get\_user())

current\_user并不是提前准备好的或是计算好的，而是赋予了它一些回调函数，在current\_user访问它的属性的时候，就触发了一系列的计算。\_get\_user()就相当于是它的回调函数。flask是在werkzeug的基础上构建的,werkzeug是一个遵循WSGI规范的HTTP开发包。

//所以一般在before\_request里面把current\_user保存下来。

LocalProxy的功能暂不深究.

## 4.2 flask\_login.py

### 4.2.1 类LoginManager

**class** **LoginManager**(object):

#### 4.2.1.1 函数\_\_init\_\_

**def** **\_\_init\_\_**(*self*, app=**None**, add\_context\_processor=**True**):

#: A class or factory function that produces an anonymous user, which

#: is used when no one is logged in.

*self*.anonymous\_user = AnonymousUserMixin

#: The name of the view to redirect to when the user needs to log in.

#: (This can be an absolute URL as well, if your authentication

#: machinery is external to your application.)

*self*.login\_view = **None**

#: The message to flash when a user is redirected to the login page.

*self*.login\_message = LOGIN\_MESSAGE

#### 4.2.1.2 函数init\_app

**def** **init\_app**(*self*, app, add\_context\_processor=**True**):

'''

Configures an application. This registers an `after\_request` call, and

attaches this `LoginManager` to it as `app.login\_manager`.

**:param** app: The :class:`flask.Flask` object to configure.

**:type** app: :class:`flask.Flask`

**:param** add\_context\_processor: Whether to add a context processor to

the app that adds a `current\_user` variable to the template.

Defaults to ``True``.

**:type** add\_context\_processor: bool

'''

app.login\_manager = *self*

app.after\_request(*self*.\_update\_remember\_cookie)

*self*.\_login\_disabled = app.config.get('LOGIN\_DISABLED',

app.config.get('TESTING', **False**))

**if** add\_context\_processor:

app.context\_processor(\_user\_context\_processor)

### 4.2.2 函数\_get\_user

**def** **\_get\_user**():

**if** **not** hasattr(\_request\_ctx\_stack.top, 'user'):

current\_app.login\_manager.\_load\_user()

**return** getattr(\_request\_ctx\_stack.top, 'user', **None**)

### 4.2.3 函数\_load\_user

**def** **\_load\_user**(*self*):

'''Loads user from session or remember\_me cookie as applicable'''

user\_accessed.send(current\_app.\_get\_current\_object())

# first check SESSION\_PROTECTION

config = current\_app.config

**if** config.get('SESSION\_PROTECTION', *self*.session\_protection):

deleted = *self*.\_session\_protection()

**if** deleted:

**return** *self*.reload\_user()

is\_missing\_user\_id = 'user\_id' **not** **in** session

**if** is\_missing\_user\_id:

cookie\_name = config.get('REMEMBER\_COOKIE\_NAME', COOKIE\_NAME)

header\_name = config.get('AUTH\_HEADER\_NAME', AUTH\_HEADER\_NAME)

has\_cookie = (cookie\_name **in** request.cookies **and**

session.get('remember') != 'clear')

**if** has\_cookie:

**return** *self*.\_load\_from\_cookie(request.cookies[cookie\_name])

**elif** header\_name **in** request.headers:

**return** *self*.\_load\_from\_header(request.headers[header\_name])

**return** *self*.reload\_user()

session里面如果有user\_id就认为是登录的。

### 4.2.4 函数\_load\_from\_cookie

**def** **\_load\_from\_cookie**(*self*, cookie):

**if** *self*.token\_callback:

user = *self*.token\_callback(cookie)

**if** user **is** **not** **None**:

session['user\_id'] = getattr(user, *self*.id\_attribute)()

session['\_fresh'] = **False**

\_request\_ctx\_stack.top.user = user

**else**:

*self*.reload\_user()

**else**:

user\_id = decode\_cookie(cookie)

**if** user\_id **is** **not** **None**:

session['user\_id'] = user\_id

session['\_fresh'] = **False**

*self*.reload\_user()

**if** \_request\_ctx\_stack.top.user **is** **not** **None**:

app = current\_app.\_get\_current\_object()

user\_loaded\_from\_cookie.send(app, user=\_get\_user())

可以从remember\_token来加载用户

# 5. 全局变量

## 5.1 session

Flask的session通过cookie来保存。而没有存储在服务器端。

Flask的底层会进行session的解析。

### 5.1.1 session解析

*self*.session = *self*.app.open\_session(*self*.request)

flktrace.flask\_trace(["self.session:", *self*.session])

**if** *self*.session **is** **None**:

*self*.session = *self*.app.make\_null\_session()

### 5.1.2 session过期

httponly = *self*.get\_cookie\_httponly(app)

secure = *self*.get\_cookie\_secure(app)

expires = *self*.get\_expiration\_time(app, session)

如果Cookie的过期属性设置为Session，那么当浏览器关闭时，这个Cookie就过期了，这里的Session就使用了这种策略。如果Cookie的expires属性不设置，就默认为Session属性。即浏览器关闭时，Cookie过期。所以下次登录时，这个session就不起作用了。

# 6.Flask源码

## 6.1 app.py

### 6.1.1 类Flask

#### 6.1.1.1 函数\_\_init\_\_

#### 6.1.1.2 函数add\_url\_rule

@setupmethod

**def** **add\_url\_rule**(*self*, rule, endpoint=**None**, view\_func=**None**, \*\*options):

**if** endpoint **is** **None**:

endpoint = \_endpoint\_from\_view\_func(view\_func)

options['endpoint'] = endpoint

methods = options.pop('methods', **None**)

下面是关键代码。

rule = *self*.url\_rule\_class(rule, methods=methods, \*\*options)

rule.provide\_automatic\_options = provide\_automatic\_options

*self*.url\_map.add(rule)

**if** view\_func **is** **not** **None**:

old\_func = *self*.view\_functions.get(endpoint)

**if** old\_func **is** **not** **None** **and** old\_func != view\_func:

**raise** AssertionError('View function mapping is overwriting an '

'existing endpoint function: %s' % endpoint)

flkdebug.trace('flask',["add\_url\_rule:endpoint,view\_func:", endpoint, view\_func])

*self*.view\_functions[endpoint] = view\_func

# 7. Werkzeug

## 7.1 LocalProxy

提供一个变量的代理的功能。

## 7.2 routing.py

### 7.2.1 类Map

### 7.2.1 类Rule

# 附录1

# 附录2