# OAuth开发流程

## 安装开发环境

安装ruby、rails、mysql

### 安装ruby

Ruby版本ruby-1.8.7,官网下载，windows环境下双击安装，自带rubygem，安装完ruby后rubygams一同安装好，gem version 1.5.2

### 安装rails

Rails版本选择2.3.5

gem install rails –version=2.3.5

### 安装MySql

gem install mysql

gem install ruby-mysql

将mysql安装目录/lib里的libmysql.dll考到ruby安装目录/bin下

修改rails项目中的database.yml问将连接数据库

或者修改rails的默认数据库，使rails在生成项目是默认使用mysql数据库

目录：ruby安装目录/lib/ruby/gems/1.8/rails\_2.0.2/lib/rails\_generators/applications/app

文件：app\_generator.rb

行数：10

将DEFAULT\_DATABASE = 'sqlite3' 改为：DEFAULT\_DATABASE = 'mysql'

## 安装restful\_authentication

gem install restful\_authentication

将 resful\_authentication文件夹拷贝到项目目录vendors/plugin/下

生成框架代码

ruby script/generator authenticated user session

在application\_controler.rb中添加：

include AuthenticatedSystem

require "oauth/consumer"

require "oauth/server"

## 安装oauth库和插件

网络安装方法：

gem install oauth

gem install oauth-plugin

本地安装方法：

版本选择：oauth-0.3.6.gem；oauth-plugin-0.3.14.gem

首先需要安装ruby-hmac

gem install ruby-hmac

gem install oauth-0.3.6.gem –local

gem install oauth-plugin-0.3.14.gem --local

在environment.rb文件中添加以下内容：

if Gem::VERSION >= "1.3.6"

module Rails

class GemDependency

def requirement

r = super

(r == Gem::Requirement.default) ? nil : r

end

end

end

end

config.gem "oauth"

config.gem "oauth-plugin"

## 生成oauth provider

ruby script/generate oauth\_provider

在项目的modle/user下加入如下内容：

has\_many :client\_applications

has\_many :tokens, :class\_name=>"OauthToken",:order=>"authorized\_at desc",:include=>[:client\_application]

迁移数据库：

rake db:migrate

至此，OAuth认证模块安装完成，可以注册用户、注册应用、编写client端代码测试获取request\_token、获取access\_token、获取资源，获取资源时需要在要获取资源的文件头部添加：

before\_filter :oauth\_required

测试成功后在进行一下步骤。

## 修改OAuth模块部分文件代码

根据智慧北邮平台的需要，需要修改部分代码实现和平台的整合连接。

平台需求：

1. 用户登录认证部分要求采用平台用户数据库，不能采用OAuth的Authentication模块的用户数据库，开源库本身使用的是OAuth自带的数据库，需修改。
2. 在将OAuth和visual sensor整合到一起的时，用于OAuth和visual sensor都带有用户登录认证部分，需要将两个认证部分合并，以便实现用户在用在二者之一登录即在对方登录，在二者之一登出即在对方登出。
3. 过滤器规则修改。在需要OAuth认证才能获取的资源controller中需要在controller的头部加上before\_filter :login\_or\_oauth\_required或before\_filter :oauth\_required，这样就使整个controller的资源需要通过OAuth和登录才能获取，然而visual sensor存在三种资源的访问方式：无需登录获取、登录获取、OAuth获取，加上过滤器后三种方式产生矛盾，需解决。

### 连接智慧北邮平台用户数据库

修改文件app/models/user.rb

添加：

require ‘rest\_client’

require ‘json’

修改authenticate(login,password)方法，修改成以下内容：

def self.authenticate(login, password)

return nil if login.blank? || password.blank?

# 登录接口 获取用户登录的session和用户id

begin

response = RestClient.post 'oray507.gicp.net:8892'+'/session',{'session[app\_name]' => 'you','session[app\_password]'=>'you\_mobroad','session[username]' => login,'session[password]'=>password}

@user= JSON.parse response.body #获取的json解析方式

userid = @user['entry']['user\_id'] #用户id

unless find\_by\_login(userid) then

uinfo={"password\_confirmation"=>"123456","login"=> userid,"password"=>"123456","email"=>userid+"@gmail.com"}

User.new(uinfo).save

end

u = find\_by\_login(userid)

return u

rescue => e

return nil

end

end

修改方法，由于oauth自带的用户数据表项库和从平台获得数据内容不太一样，这里能用到的只有usrid,相当于oauth user表里的login，修改方法为，保留oauth 中的user表，用户输入用户名密码后将其送到平台进行验证，诺认证通过则返回userid，否则返回错误，根据返回的userid查找user表里是否存在userid（即login项），诺存在则返回整条查询记录（u = find\_by\_login(userid)），诺不存在则新建一条记录：

unless find\_by\_login(userid) then

uinfo={"password\_confirmation"=>"123456","login"=> userid,"password"=>"123456","email"=>userid+"@gmail.com"}

User.new(uinfo).save

end

此条记录中只有login有用，其他项可以伪造，同样返回给函数整条查询记录（u）。

修改完成后使用you的账户进行测试。

### 整合OAuth和visual sensor

方法一：按照前面的方法在visual sensor项目（RRE）里安装生成OAuth；

方法二：将OAuth项目中的文件拷贝到visual sensor相对应的文件夹下，建议，restful\_authentication部分采用生成的方法，拷贝文件的话某些文件里缺少内容，手动修改起来比较麻烦。

完成这步后由于OAuth和visual sensor都有各自的用户登录系统，前者命名为sessions，后者命名为session，登录时页面跳转会发生混乱，需要将二者的登录部分合二为一，并且实现同时登陆同时登出的效果。

### 整合OAuth与visual sensor用户登录部分

方法：保留OAuth登录部分，将visual sensor登录部分代码添加到OAuth sessions中，删除visual sensor中的session。

第一步：将RRE/view/session下的文件拷贝到RRE/view/sessions中，删除RRE/view/session。

第二步：修改RRE/view/visualsensor/\_visual\_header.erb文件

将以下三行代码中的session改为sessions

<form action="session" method="post" id="log"

<a href="session/register">注册</a>

<label class="small\_field"> <%= link\_to "Logout", :controller =>:session, :action => :logout %></label>>

第三步：修改RRE/controller/sessions\_controller.rb文件

在类头部添加：

require 'rest\_client'

require 'json'

require 'activerecord'

将RRE/controller/session\_controller.rb文件中的所有方法拷贝到RRE/controller/sessions\_controller.rb文件中。

Create函数是OAuth认证时用户认证时调用的函数，index方法是用户在登录visual sensor时调用的函数，要实现单点登录，解决方法：调用create方法后，再调用index方法，相当于在两部分同时登陆；调用index方法后再调用create方法。为了防止循环调用，每个方法中设置标志位，判断上个方法是否调用过。

修改create方法，改为如下内容：

def create(username=nil,password=nil)

logout\_keeping\_session!

if params[:login]!=nil

username=params[:login]

password=params[:password]

sign=1

else

end

user = User.authenticate(username, password)

if user

# Protects against session fixation attacks, causes request forgery

# protection if user resubmits an earlier form using back

# button. Uncomment if you understand the tradeoffs.

# reset\_session

self.current\_user = user

new\_cookie\_flag = (params[:remember\_me] == "1")

handle\_remember\_cookie! new\_cookie\_flag

if sign==1

redirect\_back\_or\_default('/')

index(username,password)

end

flash[:notice] = "Logged in successfully"

else

note\_failed\_signin

@login = params[:login]

@remember\_me = params[:remember\_me]

render :action => 'new'

end

end

修改index方法，修改为如下内容

def index(username=nil,password=nil) #实现用户登录功能

flash[:notice]=nil

session[:usercookie]=nil

if request.post?

if username==nil||password==nil

username =params[:username]

password =params[:password]

sign=1

end

#登录部分代码

begin

# 登录接口 获取用户登录的session和用户id

response = RestClient.post COS\_URL+'/session',{'session[app\_name]' => 'you','session[app\_password]'=>'you\_mobroad','session[username]' => username,'session[password]'=>password}

@user= JSON.parse response.body #获取的json解析方式

session[:usercookie]=response.cookies

session[:userid]=@user['entry']['user\_id']

# 根据用户id获取个人信息 个人信息中有用户名之类的信息用于显示

get\_userinfo

if sign==1

create(username,password)

end

rescue =>e

flash[:notice]=e.to\_str

end

end

end

## 附录

### OAuth模块数据库内容介绍

OAuth模块数据库共有5张表，分别为：client\_applications、oauth\_nonces、oauth\_token、users、schema\_migration。下面分别介绍下每张表的作用和内容。

users：用户表，记录用户信息

表中项如下：

+-----------------------+----------- -+-----+-----+--------+---------------+

| Field | Type | Null | Key | Default | Extra |

+-----------------------+-------------+-----+-----+--------+---------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| login | varchar(40) | YES | UNI | NULL | |

| name | varchar(100) | YES | | | |

| email | varchar(100) | YES | | NULL | |

| crypted\_password | varchar(40) | YES | | NULL | |

| salt | varchar(40) | YES | | NULL | |

| created\_at | datetime | YES | | NULL | |

| updated\_at | datetime | YES | | NULL | |

| remember\_token | varchar(40) | YES | | NULL | |

| remember\_token\_expires\_at| datetime | YES | | NULL | |

+------------------------+-------------+------+----+--------+--------------+

Client\_applications:记录用户注册过的应用，内容如下：

+--------------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------------+--------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| name | varchar(255) | YES | | NULL | |

| url | varchar(255) | YES | | NULL | |

| support\_url | varchar(255) | YES | | NULL | |

| callback\_url | varchar(255) | YES | | NULL | |

| key | varchar(20) | YES | UNI | NULL | |

| secret | varchar(40) | YES | | NULL | |

| user\_id | int(11) | YES | | NULL | |

| created\_at | datetime | YES | | NULL | |

| updated\_at | datetime | YES | | NULL | |

+--------------+--------------+------+-----+---------+----------------+

oauth\_token：记录request\_token、access\_token等信息，内容如下：

+--------------------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------------------+-------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| user\_id | int(11) | YES | | NULL | |

| type | varchar(20) | YES | | NULL | |

| client\_application\_id | int(11) | YES | | NULL | |

| token | varchar(20) | YES | UNI | NULL | |

| secret | varchar(40) | YES | | NULL | |

| callback\_url | varchar(255) | YES | | NULL | |

| verifier | varchar(20) | YES | | NULL | |

| authorized\_at | datetime | YES | | NULL | |

| invalidated\_at | datetime | YES | | NULL | |

| created\_at | datetime | YES | | NULL | |

| updated\_at | datetime | YES | | NULL | |

+---------------------+-------------+------+-----+--------+--------------+

oauth\_nonces:记录oauth流程中用到的nonces值和timestamp值，内容如下：

+------------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+------------+--------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| nonce | varchar(255) | YES | MUL | NULL | |

| timestamp | int(11) | YES | | NULL | |

| created\_at | datetime | YES | | NULL | |

| updated\_at | datetime | YES | | NULL | |

+------------+--------------+------+-----+---------+----------------+