Code Smell

- Zhengyang Zhou

The most 2 smelly part in user service of our nanotwitter

1. the PREFIX variable is a Orphan variable, which I should just put it in service.rb instead of putting this variable alone in another file. Now this orphan variable is put into service.rb.

2. The login method is too long. I should refactor it. In detail, There are two cases in the login depends on whether we have the corresponding user record on redis. However, no matter we check the authentication through database or redis, we still need to generate the token and keep this record on redis. In this case, I come up with an idea to do a two-layer embedding. First, I have a **tokenized function** to handle token generation, which will be embedded in **database_login function** and **redis_login function**. Then, in the **post PREFIX + '/login' method**, I call the **tokenized function** and **database_login function** by condition. As a result, all of these function are less than 15 lines, and easier to undertand.

Here is the Before and After
Before

```
post PREFIX + '/login' do
  first_try = JSON.parse($redis.get params['username'])
  if (first_try
    && BCrypt::Password.new(first_try["password"]) == params['username'])
    user_hash = Hash.new
    user_hash["id"] = first_try["id"]
    user_hash["username"] = params['username']
    token = SecureRandom.hex
    $redis.set token, user_hash.to_json
    $redis.expire token, 432000
    u_hash = JSON.parse($redis.get(first_try["id"]))
    u_hash['leaders'] = $redis_follow.get(first_try["id"].to_s + ' leaders')
    if !u_hash['leaders']
        leader_link =
        follow_service + "#{token}/users/#{first_try["id"].to_s}/leader-list"
        u_hash['leaders'] = JSON.parse(RestClient.get leader_link, {})
    end
    return {user: u_hash, token: token}.to_json
  else
    @user = User.find_by_username(params['username'])
    if !@user.nil? && @user.password == params['password']
      token = SecureRandom.hex
      user_hash = Hash.new
      user_hash["id"] = @user.id
```

```
        user_hash["username"] = @user.username
        $redis.set token, user_hash.to_json
        $redis.expire token, 432000
        u_hash = @user.as_json
        leader_link =
        follow_service + "#{token}/users/#{first_try["id"].to_s}/leader-list"
        u_hash['leaders'] = JSON.parse(RestClient.get leader_link, {})
        return {user: u_hash, token: token}.to_json
      end
    end

    {err: true}.to_json
end
```

After

```
# Generate the token for loggined user
def tokenized(user_hash,token,id,username)
  user_hash["id"] = id
  user_hash["username"] = username
  $redis.set token, user_hash.to_json
  $redis.expire token, 432000
end

# login when there is no record on redis, we have to turn to database
def database_login(user,follow_service)
  token = SecureRandom.hex
  user_hash = Hash.new
  tokenized(user_hash,token,user.id,user.username)
  u_hash = user.as_json
  leader_link =  follow_service + "#{token}/users/#{user.id.to_s}/leader-list"
  u_hash['leaders'] = JSON.parse(RestClient.get leader_link, {})
  return {user: u_hash, token: token}
end

# login when there is record on redis
def redis_login(id,username,follow_service)
  user_hash = Hash.new
  token = SecureRandom.hex
  tokenized(user_hash,token,id,username)
  u_hash = JSON.parse($redis.get(id))
  u_hash['leaders'] = $redis_follow.get(id.to_s + ' leaders')
  if !u_hash['leaders']
      leader_link =  follow_service + "#{token}/users/#{id.to_s}/leader-list"
      u_hash['leaders'] = JSON.parse(RestClient.get leader_link, {})
  end
  return {user: u_hash, token: token}
end

# handle the login depends on whether there is record on redis
post PREFIX + '/login' do
  first_try = JSON.parse($redis.get params['username'])
```

```
    if (first_try &&
      BCrypt::Password.new(first_try["password"]) == params['password'])
      result = redis_login(first_try["id"],params['username'],follow_service)
      return result.to_json
    else
      @user = User.find_by_username(params['username'])
      if !@user.nil? && @user.password == params['password']
        result = database_login(@user,follow_service)
        return result.to_json
      end
    end
    {err: true}.to_json
  end
```