django-rest-auth Documentation

Release 0.3.0

Tivix Inc.

October 13, 2014

Contents

Warning: Version 0.3.0 is not compatible with any of previous versions, see *Changelog* section for a list of changes.

CHAPTER 1

Contents

1.1 Introduction

Since the introduction of django-rest-framework, Django apps have been able to serve up app-level REST API endpoints. As a result, we saw a lot of instances where developers implemented their own REST registration API endpoints here and there, snippets, and so on. We aim to solve this demand by providing django-rest-auth, a set of REST API endpoints to handle User Registration and Authentication tasks. By having these API endpoints, your client apps such as AngularJS, iOS, Android, and others can communicate to your Django backend site independently via REST APIs for User Management. Of course, we'll add more API endpoints as we see the demand.

1.1.1 Features

- User Registration with activation
- Login/Logout
- Retrieve/Update the Django User model
- Password change
- Password reset via e-mail
- Social Media authentication

1.1.2 Apps structure

- `rest_auth` has basic auth functionality like login, logout, password reset and password change
- `rest_auth.registration` has logic related with registration and social media authentication

1.2 Installation

1. Install package:

```
pip install django-rest-auth
```

2. Add `rest_auth` app to INSTALLED_APPS in your django settings.py:

```
INSTALLED_APPS = (
    ...,
    'rest_framework',
    'rest_framework.authtoken',
    ...,
    'rest_auth'
)
```

Note: This project depends on `django-rest-framework` library, so install it if you haven't done yet. Make sure also you have installed `rest_framework` and `rest_framework.authtoken` apps

3. Add rest_auth urls:

```
urlpatterns = patterns('',
    ...,
    (r'^rest-auth/', include('rest_auth.urls'))
)
```

You're good to go now!

1.2.1 Registration (optional)

1. If you want to enable standard registration process you will need to install `django-allauth` - see this doc for installation http://django-allauth.readthedocs.org/en/latest/installation.html.
2. Add `allauth`, `allauth.account` and `rest_auth.registration` apps to INSTALLED_APPS in your django settings.py:

```
INSTALLED_APPS = (
    ...,
    'allauth',
    'allauth.account',
    'rest_auth.registration',
)
```

3. Add rest_auth.registration urls:

```
urlpatterns = patterns('',
    ...,
    (r'^rest-auth/', include('rest_auth.urls'))
    (r'^rest-auth/registration/', include('rest_auth.registration.urls'))
)
```

1.2.2 Social Authenitcation (optional)

Using `django-allauth`, `django-rest-auth` provides helpful class for creating social media authentication view. Below is an example with Facebook authentication.

Note: Points 1, 2 and 3 are related with `django-allauth` configuration, so if you have already configured social authentication, then please go to step 4. See `django-allauth` documentation for more details.

1. Add `allauth.socialaccount` and `allauth.socialaccount.providers.facebook` apps to INSTALLED_APPS in your django settings.py:

```
INSTALLED_APPS = (
    ...,
    'rest_framework',
    'rest_framework.authtoken',
    'rest_auth'
    ...,
    'allauth',
    'allauth.account',
    'rest_auth.registration',
    ...,
    'allauth.socialaccount',
    'allauth.socialaccount.providers.facebook',
)
```

2. Add `allauth.socialaccount.context_processors.socialaccount` to TEMPLATE_CONTEXT_PROCESSORS in django settings
3. Add Social Application in django admin panel
4. Create new view as a subclass of `rest_auth.registration.views.SocialLogin` with `FacebookOAuth2Adapter` adapter as an attribute:

```
from allauth.socialaccount.providers.facebook.views import FacebookOAuth2Adapter
from rest_auth.registration.views import SocialLogin

class FacebookLogin(SocialLogin):
    adapter_class = FacebookOAuth2Adapter
```

5. Create url for FacebookLogin view:

```
urlpatterns += pattern('',
    ...,
    url(r'^/rest-auth/facebook/$', FacebookLogin.as_view(), name='fb_login')
)
```

1.3 API endpoints

1.3.1 Basic

- /rest-auth/login/ (POST)
  - username (string)
  - password (string)
- /rest-auth/logout/ (POST)
- /rest-auth/password/reset/ (POST)
  - email
- /rest-auth/password/reset/confim/ (POST)
  - uid
  - token
  - new_password1

    - new_password2

Note: uid and token are sent in email after calling /rest-auth/password/reset/

- /rest-auth/password/change/ (POST)
  - new_password1
  - new_password2
- /rest-auth/user/ (GET)
- /rest-auth/user/ (PUT/PATCH)
  - username
  - first_name
  - last_name
  - email

1.3.2 Registration

- /rest-auth/registration/ (POST)
  - username
  - password1
  - password2
  - email

Note: This endpoint is based on `allauth.account.views.SignupView` and uses the same form as in this view. To override fields you have to create custom Signup Form and define it in django settings:

```
ACCOUNT_FORMS = {
    'signup': 'path.to.custom.SignupForm'
}
```

See allauth documentation for more details.

- /rest-auth/registration/ (POST)
  - key

1.3.3 Social Media Authentication

Basing on example from installation section *Installation*

- /rest-auth/facebook/ (POST)
  - access_token

1.4 Configuration

- **REST_AUTH_SERIALIZERS**

  You can define your custom serializers for each endpoint without overriding urls and views by adding `REST_AUTH_SERIALIZERS` dictionary in your django settings. Possible key values:

  - LOGIN_SERIALIZER - serializer class in `rest_auth.views.Login`, default value `rest_auth.serializers.LoginSerializer`
  - TOKEN_SERIALIZER - response for successful authentication in `rest_auth.views.Login`, default value `rest_auth.serializers.TokenSerializer`
  - USER_DETAILS_SERIALIZER - serializer class in `rest_auth.views.UserDetails`, default value `rest_auth.serializers.UserDetailsSerializer`
  - PASSWORD_RESET_SERIALIZER - serializer class in `rest_auth.views.PasswordReset`, default value `rest_auth.serializers.PasswordResetSerial`
  - PASSWORD_RESET_CONFIRM_SERIALIZER - serializer class in `rest_auth.views.PasswordResetConfirm`, default value `rest_auth.serializers.PasswordResetConfirmSerializer`
  - PASSWORD_CHANGE_SERIALIZER - serializer class in `rest_auth.views.PasswordChange`, default value `rest_auth.serializers.PasswordChangeSeri`

  Example configuration:

  ```
  REST_AUTH_SERIALIZERS = {
      'LOGIN_SERIALIZER': 'path.to.custom.LoginSerializer',
      'TOKEN_SERIALIZER': 'path.to.custom.TokenSerializer',
      ...
  }
  ```

- **REST_SESSION_LOGIN** - Enable session login in Login API view (default: True)

1.5 Changelog

1.5.1 0.3.0

- replaced `django-registration` with `django-allauth`
- moved registration logic to separated django application (`rest_auth.registration`)
- added serializers customization in django settings
- added social media authentication view
- changed request method from GET to POST in logout endpoint
- changed request method from POST to PUT/PATCH for user details edition
- changed password reset confim url - uid and token should be sent in POST
- increase test coverage
- made compatibile with django 1.7
- removed user profile support