aruseni May 27, 2013 at 01:31 PM

Django: Using QR codes to quickly access a sit
devices

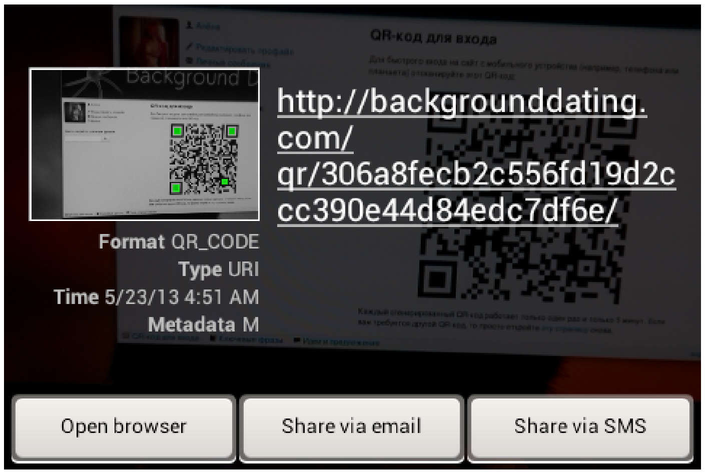
Website development , python , Django

Tutorial

If you have a site that is often used from mobile devices (such as phones and tablets), then
implement quick login - so that the user does not need to enter either the website address
password).

On some sites, you may have seen the ability to send an SMS message with a link for quick
thing. The main difference of the approach described in this note is that instead of sending
code that contains a link that allows you to enter the site without entering authorization data

Ads



By the way, the whole process of writing the application, which is given below, can be viewed in the screencast (available on YouTube , or in better quality asMPEG2 file in 1080p).

Before you begin to implement this option of authorization, let's consider how it differs from the option with sending an SMS message:

- An SMS message is a little safer: on the one hand, we transmit the link through third parties (in particular, the SMS gate, the mobile operator), but on the other hand, firstly, the link will be unavailable via JS (and, accordingly, even if the site has XSS, then the attacker will not be able to get the link from the SMS message - but you can get it from the QR code), and secondly, get physical access to the computer with a browser in which the user is authorized on the site, in general easier than accessing a computer, and at the same time, to the body user background
- An SMS message, at the same time, is not so universal: it will not work if for some reason there is no cellular connection, or the mobile device used by the user does not imply the possibility of working in cellular networks (but it also has a camera, and Internet connection, and an application for scanning QR codes on it is or is easily installed)
- Sending SMS messages costs money; generating QR codes is absolutely free
- SMS messages in some situations can significantly drop speed and reliability indicators (that is, a message may arrive with a significant delay, or even may not arrive at all), and scanning QR codes is predictable and usually works well (at least if the camera is all right)

- A site requires a user's phone number to send SMS messages, but not all users like the idea of entering their phone number somewhere on the Internet

It turns out that the option with QR codes is very good. And even if high security is important to us, theoretically no one bothers sending a QR code by e-mail, or, for example, asking for a password each time (once again entering a password on a convenient large computer keyboard, it seems to me a lot easier than entering a website address + username / email address + password on the virtual keyboard of the mobile device). Nevertheless, now I propose to implement the simplest, most basic option for quick entry using QR codes, the implementation of which will take us a minimum of time.

The text below, like the screencast, explains various implementation features. If you want to quickly add the application to the site, then you can install it through pip:

```
pip install django-qrauth
```

In this case, you only need to include the urls scheme of the application in your main urls.py, as well as add templates. You can find installation instructions as well as the source code on Github .

The following describes how you can build the application yourself - relevant, for example, if you immediately want to edit something in it.

So, first of all, we will go to the working directory of the Django-project, in which we want to add such authorization, and create a new application. Call it, for example, qrauth:

```
python manage.py startapp qrauth
```

In the directory that appears, create the qr.py file:

```
try:
    from PIL import Image, ImageDraw
except ImportError:
    import Image, ImageDraw

import qrcode.image.base
import qrcode.image.pil

class PilImage(qrcode.image.pil.PilImage):
    def __init__(self, border, width, box_size):
        if Image is None and ImageDraw is None:
            raise NotImplementedError("PIL not available")
        qrcode.image.base.BaseImage.__init__(self, border, width, box_size)
        self.kind = "PNG"

        pixelsize = (self.width + self.border * 2) * self.box_size
        self._img = Image.new("RGBA", (pixelsize, pixelsize))
        self._idr = ImageDraw.Draw(self._img)

def make_qr_code(string):
    return qrcode.make(string, box_size=10, border=1, image_factory=PilImage)
```

The python-qrcode module is used here . You can install it using pip:

```
pip install qrcode
```

In order to get pictures with a transparent (rather than white) background, we specially use our class to create pictures, inheriting it from qrcode.image.pil.PilImage. If you are satisfied with pictures with a white background, then it will be enough to write like this:

```
import qrcode

def make_qr_code(string):
    return qrcode.make(string, box_size=10, border=1)
```

It is worth noting that in this case, the pictures that qrcode.make returns (and, accordingly, the make_qr_code function) are not optimal in terms of size. For example, using optipng, their size can be reduced by about 70% (of course, without loss of quality). However, in most cases this is not fundamental - in any case, their size is small (within a few kibytes).

Next, create the utils.py file and add the functions that we will then use in the views:

```
import os
import string
import hashlib

from django.conf import settings

def generate_random_string(length,
                           stringset="".join(
                               [string.ascii_letters+string.digits]
                           )):
    """
    Returns a string with `length` characters chosen from `stringset`
    >>> len(generate_random_string(20) == 20
    """
    return "".join([stringset[i%len(stringset)] \
        for i in [ord(x) for x in os.urandom(length)]])

def salted_hash(string):
    return hashlib.sha1(":)".join([
        string,
        settings.SECRET_KEY,
    ])).hexdigest()
```

The generate_random_string function generates a string of random characters of a given length. By default, the string is composed of letters of the Latin alphabet (both lower and upper case) and numbers.

The salted_hash function salt and hash the string.

Now open views.py and write the views:

```
import redis

from django.contrib.auth.decorators import login_required
from django.contrib.auth import login, get_backends
from django.contrib.sites.models import get_current_site
from django.template import RequestContext
from django.shortcuts import render_to_response
from django.http import HttpResponse, HttpResponseRedirect, Http404
from django.core.urlresolvers import reverse

from django.contrib.auth.models import User

from privatemessages.context_processors import \
number_of_new_messages_processor

from utils import generate_random_string, salted_hash
from qr import make_qr_code

@login_required
def qr_code_page(request):
    r = redis.StrictRedis()

    auth_code = generate_random_string(50)
    auth_code_hash = salted_hash(auth_code)

    r.setex(auth_code_hash, 300, request.user.id)

    return render_to_response("qrauth/page.html",
                              {"auth_code": auth_code},
                              context_instance=RequestContext(request))

@login_required
def qr_code_picture(request, auth_code):
    r = redis.StrictRedis()

    auth_code_hash = salted_hash(auth_code)

    user_id = r.get(auth_code_hash)

    if (user_id == None) or (int(user_id) != request.user.id):
        raise Http404("No such auth code")

    current_site = get_current_site(request)
    scheme = request.is_secure() and "https" or "http"

    login_link = "".join([
        scheme,
        "://",
        current_site.domain,
        reverse("qr_code_login", args=(auth_code_hash,)),
    ])

    img = make_qr_code(login_link)
    response = HttpResponse(mimetype="image/png")
    img.save(response, "PNG")
    return response

def login_view(request, auth_code_hash):
    r = redis.StrictRedis()
    user_id = r.get(auth_code_hash)

    if user_id == None:
        return HttpResponseRedirect(reverse("invalid_auth_code"))

    r.delete(auth_code_hash)

    try:
        user = User.objects.get(id=user_id)
    except User.DoesNotExist:
        return HttpResponseRedirect(reverse("invalid_auth_code"))

    # In lieu of a call to authenticate()
    backend = get_backends()[0]
    user.backend = "%s.%s" % (backend.__module__, backend.__class__.__name__)
    login(request, user)

    return HttpResponseRedirect(reverse("dating.views.index"))
```

When accessing a page with a QR code (qr_code_page), a random string of 50 characters is generated. Then, in Redis (you can install the client using `pip install redis`), a new key-value pair is added, where the salt hash of the generated random string is set as the key, and the user ID is set as the value (this is necessary so that the picture with the QR code is added to the page, was accessible only to this user). This key sets the expiration time, in the example 300 seconds (5 minutes) are indicated.

In the context of the template, the generated random string is set: this line is then used in the address to which the picture with the QR code is returned (but for the authorization, the hash is used, and the address with the hash is included in the QR code: thus, even if someone else knows the address of the picture, then for authorization this will not be enough - for authorization you need to know the salty hash for a random string specified in the address of the picture).

Further, when loading the image (qr_code_picture), the random string contained in the address of the image is hashed again, and then it is checked whether Redis has a corresponding key. If there is such a key and contains the identifier of the current user, then a QR code is created and returned containing an absolute link for instant authorization on the site. Otherwise, error 404 is returned.

By the way, you understand what can be easily improved here?

Obtaining a domain here is done using django.contrib.sites . You can specify a domain through the administrative interface (/ admin / sites / site /).

If your server is behind a reverse proxy (for example, nginx) and you use SSL, then make sure that information about this is included in the requests to the upstream server - this is necessary so that request.is_secure () returns the correct value (for this define in the settings SECURE_PROXY_SSL_HEADER, but keep in mind that you will need to set / remove this header on the proxy server side - otherwise, if, for example, your site is accessible via HTTP and HTTPS, then a user who accesses via HTTP will be able to set this header like this so that request.is_secure () will return True, which is bad from a security point of view).

And yes, starting in Python 2.6 `request.is_secure() and "https" or "http"`you can write instead `"https" if request.is_secure() else "http"`.

When clicking on the link for instant authorization, it is checked whether Redis has a key corresponding to the hash specified in the link. If not, the user is redirected to a page with a message stating that the QR code is invalid (in this example, this page does not require a separate submission). If there is, then the key in Redis is deleted, after which it is checked whether there is a user with this identifier in the database. If not, then again, there is a redirect to the page with a message that the QR code is invalid. If there is, then authorization occurs and the user is redirected to the main page of the site.

Now add the urls.py file and define the URL scheme of the application in it:

```
from django.conf.urls import patterns, url
from django.views.generic.simple import direct_to_template

urlpatterns = patterns('qrauth.views',
    url(
        r'^pic/(?P<auth_code>[a-zA-Z\d]{50})/$',
        'qr_code_picture',
        name='auth_qr_code'
    ),
    url(
        r'^(?P<auth_code_hash>[a-f\d]{40})/$',
        'login_view',
        name='qr_code_login'
    ),
    url(
        r'invalid_code/$',
        direct_to_template,
        {'template': 'qrauth/invalid_code.html'},
        name='invalid_auth_code'
    ),
    url(
        r'^$',
        'qr_code_page',
        name='qr_code_page'
    ),
)
```

Also, do not forget to open your main urls.py (which is indicated in ROOT_URLCONF), and include urlpatterns from urls.py of the created application there:

```
urlpatterns = patterns('',
    # …
    url(r'^qr/', include('qrauth.urls')),
    # …
)
```

Now open the templates directory and add the qrauth directory there.

Example for invalid_code.html:

```
{% extends "base.html" %}

{% block title %}QR-код недействителен{% endblock %}

{% block content %}
<div class="error">
    <h1>QR-код недействителен</h1>
    <p>QR-код, который вы используете для авторизации, недействителен. Пожалуйста, попробуйте ещё раз открыт
ь страницу с QR-кодом для входа и отсканировать код повторно.</p>
</div>
{% endblock %}
```

Example for page.html:

```
{% extends "base.html" %}

{% block title %}QR-код для входа{% endblock %}

{% block content %}
<div class="qr_code">
    <h1>QR-код для входа</h1>
    <p>Для быстрого входа на сайт с мобильного устройства (например, телефона или планшета) отсканируйте это
т QR-код:</p>
    <div><img src="{% url auth_qr_code auth_code %}" alt="QR"></div>
    <p>Каждый сгенерированный QR-код работает только один раз и только 5 минут. Если вам требуется другой QR
-код, то просто откройте <a href="{% url qr_code_page %}">эту страницу</a> снова.</p>
</div>
{% endblock %}
```

Actually, now it remains to open the site in a browser and check. If the QR code is successfully generated and displayed, try scanning it using your phone or something else with a camera and the Internet.

If there are any questions or thoughts about what other options may be for quick and convenient authorization from mobile devices - I will be glad to comment.

Good luck to everyone and enjoy programming!

Happy summer to you! :)

Tags: Python, Django, QR codes, QR, python-qrcode, Pil, Python Imaging Library, Web development, quick entry, instant login, fast authorization, instant authorization, QR code input, QR code authorization

+16 121 16.9k eleven Share

Arseny @aruseni
Lead Python Software Engineer

SIMILAR POSTS

October 17, 2019 at 10:00 AM

Python in Visual Studio Code - October 2019 Release

+8 1.2k 1 0

October 17, 2019 at 09:44 AM

How to Write a Smart Contract with Python on Ontology? Part 4: Native API

+5 534 1 0

October 11, 2019 at 02:18 PM

Python vs JavaScript: Which One Can Benefit You The Most?

+8 4.4k 5 3

Ads

Comments 11

voff May 27, 2013 at 01:39 PM +4

On habr articles Pts are not enough QR codes, it is necessary to throw through favorites

funca May 27, 2013 at 07:48 PM +1

there are plugins that generate qrcode (like addons.mozilla.org/en-us/firefox/addon/quickresponse/). and there's also chrome-to-phone play.google.com/store/apps/details?id=com.google.android.apps.chrometophone&hl=en .

Waxer May 27, 2013 at 10:34 PM 0

You can use a very simple and cross-browser bookmark solution .

UFO just landed and posted this here

Averrin May 27, 2013 at 02:24 PM 0

It is more interesting when you scan the code, and the page is refreshed in the browser, letting it in. I saw it in AirDroid (but there is both a server and a reader in the program itself on the device). Probably, if you use websockets or something like that, you can implement it on your server, and the reader will simply follow the link where you need to slip something like “Congratulations, you are logged in, now the page on your computer will reload.”

aruseni May 27, 2013 at 02:55 PM 0

Please clarify what is the point here if nothing is checked on the side of the mobile phone (after all, you can recognize a QR code on a computer and follow this link in the same way)? Will authorization be available to anyone who can scan a QR code from any device? Or all the same, it will be necessary to scan a special application that will additionally check something and, for example, send a POST request containing some additional parameters (for example, the session identifier that the program previously received by asking the user for a username and password and by sending an authorization request to the server)?

Averrin May 27, 2013 at 03:36 PM 0

The right questions. For AirDroid, the functionality is obvious: you must confirm that it is you who owns the device you want to enter.
For the desktop, the login option is clear as a login / password change will not work, but maybe there are some other options: confirmation, some other functionality that requires a “complicated” link click.
But I’m so, out of obstinacy, having thought, I admit that it is interesting technically, but probably useless.

Mithgol May 27, 2013 at 03:23 PM 0

For some reason, I don’t understand the task “how to implement a quick login - so that the user does not need to enter either the website address or login and password”. If the user has not identified himself (did not enter at least a login), then how does the site know **whose** input QR code to show? If the user did not enter the site address, then where (and where) does the QR code come from ?

Or is the task like this: the user accessed the site (logged in) from a regular computer - and then they show him such a QR code with which he can log in at the same time on his mobile phone (or tablet)?

Or the task is this: go to the site once, then get a QR-code, print it - and make all subsequent visits through scanningthis QR code?

aruseni May 27, 2013 at 03:43 PM +1

The second option described most closely matches the approach described in this blog post. With the exception that in this example, you can apply for the QR code after entering the site an unlimited number of times (as long as the user is authorized on the site).

That is, the point here is that it is convenient to enter the site on a computer, but not very convenient on mobile devices (for many, for example, the password contains words in Russian typed in the English layout). So, it makes sense to give users the opportunity to enter the site without manually entering the site address and any authorization data.

autobusiness May 27, 2013 at 06:08 PM -1

There is already a similar implemented QR code

aruseni May 27, 2013 at 06:13 PM +1

What harsh SEO they have at home.

PS And where are the QR codes for instant authorization?

Only users with full accounts can post comments. Log in , please.

TOP POSTS

Day Week Month

How we learned to draw text on HTML5 Canvas

+2 512 4 0

Full disclosure: 0day vulnerability (backdoor) in firmware for HiSilicon-based DVRs, NVRs and IP cameras

+17 46.8k 4 14

Audio over Bluetooth: most detailed information about profiles, codecs, and devices

+22 63k 7 8

Implement stories in the Flutter app

+2 199 2 0

Prettier is a Must-Have for Large-Scale Projects: Spent 20 Minutes Setting It Up and Forgot About Formatting for a Year

+8 2.3k 5 0

Your account	Sections	Info	Services
Log in	Posts	Rules	Ads