Working with URL, GET, POST parameters in Django

Posted by: bhaskar (/blog/author/bhaskar/) 5 years, 5 months ago

(8 Comments (/blog/working-with-url-get-post-parameters-in-django/#disqus_thread))
This tutorial tries to clarify on how one can pass parameters from one page to another page in Django.
I follow three ways:

```
1. Parameters as part of url
2. GET parameters
3. POST parameters
```

1. Parameters as part of url:

In general sense, if we want to display a list of users, the proper url can be:

```
/users
```

If we want to display the details of a particular user with id 5, proper url will be like:

```
/users/5
```

This can be achieved by setting url like this

```
url(r'^users/(?P<user_id>\d+)/$', 'viewname', name='urlname')
```

If you observe well, we have used `user_id` in the above link. This url says that the url is expecting a parameter `user_id` of type number. In regex (http://en.wikipedia.org/wiki/Regular_expression), \d is [0-9]. This user_id is passed to the view as a parameter.

```
def viewname(request, user_id):
    user = User.objects.get(id=user_id)
    #do something with this user
```

In class based view, we can get url parameter in kwargs

```
class SampleView(TemplateView):
    def get_context_data(self, **kwargs):
        user = User.objects.get(id=kwargs['user_id'])
        #do something with this user
```

2. GET parameters

First have a look at this url. One place where we use GET parameters is filters.

```
/products?price_lte=5000
```

For this kind of url, we need not specify any regex for price_lte(parameter). Url can be:

```
url(r'^products/$', 'viewname', name='urlname')

def viewname(request):
    price_lte = request.GET['price_lte']
    #Code to filter products whose price is less than price_lte i.e. 5000
```

3. POST parameters

In general case, url when sending post parameters doesnt contain anything related to parameters in the url. For example:

```
/register
```

Urls.py will contain something like

```
url(r'^register/$', 'register', name='urlname')
```

The corresponding view will be:

```
def register(request):
    form = RegisterForm()
    if request.method == "POST":
        form = RegisterForm(request.POST) #if no files
        if form.is_valid():
            #do something if form is valid
    context = {
        'form': form
    }
    return render(request, "template.html", context)
```

The above view is actually rendering the empty form on every GET request i.e. when the page is opened. On POST request, if the form is valid, it usually saves the form during the registration. If the form is invalid, it will simply render the form again with entered values and errors.
The form for the above view will be like:

```
{{ form.as_p }} #in template
```

which renders the form html.
All the html input elements having attribute `name` will be sent as POST parameters on submit. In POST parameters, value of `name` of that element will be sent as `key` and the value of that html element will be sent as value to the `key` .
For the html elements:

```
<input type='text' name='num' value=5 />
```

In view:

```
request.POST['num']  # returns 5
```

Share on Twitter (http://twitter.com/home?status=http%3A//django.cowhite.com/blog/working-with-url-get-post-parameters-in-django/%20Working%20with%20URL%2C%20GET%2C%20POST%20para

Share on Facebook (http://facebook.com/sharer.php?u=http://django.cowhite.com/blog/working-with-url-get-post-parameters-in-django/&t=Working%20with%20URL%2C%20GET%2C%20POST%20para

← Function based views in Django (/blog/function-based-views-in-django/)

Django production deployment using Ubuntu, Nginx, Gunicorn, Supervisor and Fabfile → (/blog/django-production-deployment/)

Comments

Recent Posts

Reading QR Code from Image using python (/blog/reading-qr-code-from-image-using-python/)

Open edX® - Open source MOOC platform (/blog/open-edx-open-source-mooc-platform/)

Exporting and importing data in Django using pg_dump in postgresql, mysqldump in mysql, dumpdata and loaddata commands in Django (/blog/exporting-and-importing-data-in-django-using-pg-dump-in-postgresql-mysqldump-in-mysql-dumpdata-and-loaddata-commands-in-django/)

Opensourced Django invitation system with roles and permissions (/blog/opensourced-django-invitation-system-with-roles-and-permissions/)

Using GeoDjango in your Django project (/blog/using-geodjango-in-your-django-project/)

×
Web Application Development.
We develop web applications to our customers using python/django/angular.
Contact us at hello@cowhite.com

Archive

2019

January (/blog/archive/2019/1/) (1)

2018
June (/blog/archive/2018/6/) (1)
May (/blog/archive/2018/5/) (4)

2017
August (/blog/archive/2017/8/) (3)
July (/blog/archive/2017/7/) (39)
June (/blog/archive/2017/6/) (1)
May (/blog/archive/2017/5/) (2)

2016
October (/blog/archive/2016/10/) (1)
September (/blog/archive/2016/9/) (2)
April (/blog/archive/2016/4/) (1)
February (/blog/archive/2016/2/) (2)

2015
September (/blog/archive/2015/9/) (1)

2014
October (/blog/archive/2014/10/) (3)

Tags

shallowcopy (/blog/tag/shallowcopy/) (1) node package manager (/blog/tag/node-package-manager/) (1) improve google analytics (/blog/tag/improve-google-analytics/) (1) model serializers (/blog/tag/model-serializers/) (1) middleware (/blog/tag/middleware/) (1) new middleware (/blog/tag/new-middleware/) (1) urldispatcher (/blog/tag/urldispatcher/) (2) payments (/blog/tag/payments/) (1) serializer fields (/blog/tag/serializer-fields/) (1) search ranking (/blog/tag/search-ranking/) (1) Google search console API (/blog/tag/google-search-console-api/) (1) logging (/blog/tag/logging/) (1) javascript (/blog/tag/javascript/) (1) front-end (/blog/tag/front-end/) (1) django middleware (/blog/tag/django-middleware/) (1) django example (/blog/tag/django-example/) (3) Django Rest Framework (/blog/tag/django-rest-framework/) (2) migrate (/blog/tag/migrate/) (1) django sitemap (/blog/tag/django-sitemap/) (1) django documentation (/blog/tag/django-documentation/) (4) Django Models (/blog/tag/django-models/) (1) migrations (/blog/tag/migrations/) (1) google keyword not provided (/blog/tag/google-keyword-not-provided/) (1) Database (/blog/tag/database/) (3) urlconf (/blog/tag/urlconf/) (2) full text search (/blog/tag/full-text-search/) (1) chargebee (/blog/tag/chargebee/) (1) generator (/blog/tag/generator/) (1) context manager (/blog/tag/context-manager/) (1) ranking (/blog/tag/ranking/) (1) debugging (/blog/tag/debugging/) (1) search (/blog/tag/search/) (1) deepcopy (/blog/tag/deepcopy/) (1) js dependencies (/blog/tag/js-dependencies/) (1) authentication backend (/blog/tag/authentication-backend/) (2) django (/blog/tag/django/) (13) automation (/blog/tag/automation/) (1) Derived Attributes (/blog/tag/derived-attributes/) (1) postgres (/blog/tag/postgres/) (1) django-pacakges (/blog/tag/django-pacakges/) (1) subscription (/blog/tag/subscription/) (1) tutorial (/blog/tag/tutorial/) (3) oauth toolkit (/blog/tag/oauth-toolkit/) (1) Class Property (/blog/tag/class-property/) (1) views (/blog/tag/views/) (2) oauth2 (/blog/tag/oauth2/) (1) npm (/blog/tag/npm/) (1) Security (/blog/tag/security/) (1) postgresql (/blog/tag/postgresql/) (1) full text (/blog/tag/full-text/) (1) unlock google keyword (/blog/tag/unlock-google-keyword/) (1) Dynamic fields (/blog/tag/dynamic-fields/) (1) nodejs (/blog/tag/nodejs/) (1) grunt (/blog/tag/grunt/) (1) Webhook (/blog/tag/webhook/) (1) serializers (/blog/tag/serializers/) (1) google search console (/blog/tag/google-search-console/) (1) google analytics keywords (/blog/tag/google-analytics-keywords/) (1) Python (/blog/tag/python/) (7) models (/blog/tag/models/) (1) lists (/blog/tag/lists/) (1) django rest fremework (/blog/tag/django-rest-fremework/) (1) dynamic and static sitemap (/blog/tag/dynamic-and-static-sitemap/) (1) nested lists (/blog/tag/nested-lists/) (1) django complete tutorial (/blog/tag/django-complete-tutorial/) (2) decorators (/blog/tag/decorators/) (1)

Authors

sankar (/blog/author/sankar/) (1)
bhaskar (/blog/author/bhaskar/) (13)
srinath (/blog/author/srinath/) (13)
ganesh (/blog/author/ganesh/) (11)
ravi (/blog/author/ravi/) (23)

Feeds

RSS (/blog/feeds/rss/) / Atom (/blog/feeds/atom/)

×

Web Application Development:

We develop web applications to our customers using python/django/angular.

Contact us at hello@cowhite.com