

Django - Ajax

Ajax essentially is a combination of technologies that are integrated together to reduce the number of page loads. We generally use Ajax to ease end-user experience. Using Ajax in Django can be done by directly using an Ajax library like JQuery or others. Let's say you want to use JQuery, then you need to download and serve the library on your server through Apache or others. Then use it in your template, just like you might do while developing any Ajax-based application.

Another way of using Ajax in Django is to use the Django Ajax framework. The most commonly used is django-dajax which is a powerful tool to easily and super-quickly develop asynchronous presentation logic in web applications, using Python and almost no JavaScript source code. It supports four of the most popular Ajax frameworks: Prototype, jQuery, Dojo and MooTools.

Using Django-dajax

First thing to do is to install django-dajax. This can be done using `easy_install` or `pip` –

```
$ pip install django_dajax
$ easy_install django_dajax
```

This will automatically install `django-dajaxice`, required by `django-dajax`. We then need to configure both `dajax` and `dajaxice`.

Add `dajax` and `dajaxice` in your project `settings.py` in `INSTALLED_APPS` option –

```
INSTALLED_APPS += (
    'dajaxice',
    'dajax'
)
```

Make sure in the same `settings.py` file, you have the following –

```
TEMPLATE_LOADERS = (
    'django.template.loaders.filesystem.Loader',
    'django.template.loaders.app_directories.Loader',
    'django.template.loaders.eggs.Loader',
)

TEMPLATE_CONTEXT_PROCESSORS = (
```

```

'django.contrib.auth.context_processors.auth' ,
'django.core.context_processors.debug' ,
'django.core.context_processors.i18n' ,
'django.core.context_processors.media' ,
'django.core.context_processors.static' ,
'django.core.context_processors.request' ,
'django.contrib.messages.context_processors.messages'
)

STATICFILES_FINDERS = (
'django.contrib.staticfiles.finders.FileSystemFinder' ,
'django.contrib.staticfiles.finders.AppDirectoriesFinder' ,
'dajaxice.finders.DajaxiceFinder' ,
)

DAJAXICE_MEDIA_PREFIX = 'dajaxice'

```

Now go to the myapp/url.py file and make sure you have the following to set dajax URLs and to load dajax statics js files –

```

from dajaxice.core import dajaxice_autodiscover, dajaxice_config
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.conf import settings

Then dajax urls:

urlpatterns += patterns('',
    url(r'^%s/' % settings.DAJAXICE_MEDIA_PREFIX, include('dajaxice.urls')),)

urlpatterns += staticfiles_urlpatterns()

```

Let us create a simple form based on our Dreamreal model to store it, using Ajax (means no refresh).

At first, we need our Dreamreal form in myapp/form.py.

```

class DreamrealForm( forms.Form):
    website = forms.CharField(max_length = 100)
    name = forms.CharField(max_length = 100)
    phonenumber = forms.CharField(max_length = 50)
    email = forms.CharField(max_length = 100)

```

Then we need an ajax.py file in our application: myapp/ajax.py. That's where is our logic, that's where we put the function that will be saving our form then return the popup –

```

from dajaxice.utils import deserialize_form
from myapp.form import DreamrealForm

```

```

from dajax.core import Dajax
from myapp.models import Dreamreal

@dajaxice_register
def send_form(request, form):
    dajax = Dajax()
    form = DreamrealForm(deserialize_form(form))

    if form.is_valid():
        dajax.remove_css_class('#my_form input', 'error')
        dr = Dreamreal()
        dr.website = form.cleaned_data.get('website')
        dr.name = form.cleaned_data.get('name')
        dr.phonenumber = form.cleaned_data.get('phonenumber')
        dr.save()

        dajax.alert("Dreamreal Entry %s was successfully saved." %
                    form.cleaned_data.get('name'))
    else:
        dajax.remove_css_class('#my_form input', 'error')
        for error in form.errors:
            dajax.add_css_class('#id_%s' % error, 'error')

    return dajax.json()

```

Now let's create the dreamreal.html template, which has our form –

```

<html>
  <head></head>
  <body>

    <form action = "" method = "post" id = "my_form" accept-charset = "utf-8">
      {{ form.as_p }}
      <p><input type = "button" value = "Send" onclick = "send_form();"></p>
    </form>

  </body>
</html>

```

Add the view that goes with the template in myapp/views.py –

```

def dreamreal(request):
    form = DreamrealForm()
    return render(request, 'dreamreal.html', locals())

```

Add the corresponding URL in myapp/urls.py –

```

url(r'^dreamreal/', 'dreamreal', name = 'dreamreal'),

```

Now let's add the necessary in our template to make the Ajax work –

At the top of the file add –

```
{% load static %}
{% load dajaxice_templatetags %}
```

And in the <head> section of our dreamreal.html template add –

We are using the JQuery library for this example, so add –

```
<script src = "{% static '/static/jquery-1.11.3.min.js' %}"
    type = "text/javascript" charset = "utf-8"></script>
<script src = "{% static '/static/dajax/jquery.dajax.core.js' %}"></script>
```

The Ajax function that will be called on click –

```
<script>

    function send_form(){
        Dajaxice.myapp.send_form(Dajax.process, {'form': $('#my_form').serialize(true)});
    }
</script>
```

Note that you need the “jquery-1.11.3.min.js” in your static files directory, and also the jquery.dajax.core.js. To make sure all dajax static files are served under your static directory, run –

```
$python manage.py collectstatic
```

Note – Sometimes the jquery.dajax.core.js can be missing, if that happens, just download the source and take that file and put it under your static folder.

You will get to see the following screen, upon accessing /myapp/dreamreal/ –

Using Django-dajax

On submit, you will get the following screen –

Using Django-dajax Response

Revision #1

Created 14 December 2022 11:16:43 by Admin

Updated 14 December 2022 11:17:16 by Admin