Contents

Before following this tutorial, you should understand well the basic Django tutorial and have some understanding of template inheritance, Managing static files, and Loading templates, and **ExtJs**

Prepare the project

1. Start new project

   ```
   django-admin.py startproject django_extjs_tutorial
   ```

2.
   ```
   cd django_extjs_tutorial/
   ```

3. Create two new directories in `django_extjs_tutorial/`:
   - `static/` - this will store project-wide (re-used by different apps) static files, like CSS, JavaScript. For example, we will put ExtJs library here.
   - `templates/` - this will store project-wide HTML templates. We will put here a template that includes extjs library in a header, so we don't need to repeat that in app templates.

   ```
   mkdir static
   mkdir templates
   ```

   Your directory now should look like this:

   ```
   django_extjs_tutorial/
      __init__.py
      manage.py
      settings.py
      static/
      templates/
      urls.py
   ```

4. Download ExtJs and unzip to `static/`.
5. Open `settings.py` file in a text editor.
   - On the top of the file add

     ```
     import os
     _ROOT_PATH = os.path.dirname(__file__)
     ```

     I put underscore at the beginning of the variable name just to remember that this setting has been added by me. We will re-use `_ROOT_PATH` in `settings.py` to avoid hard-coding any paths.

- Adjust `DATABASES` setting to:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3', # Add
'postgresql_psycopg2', 'postgresql', 'mysql', 'sqlite3' or
'oracle'.
        'NAME': os.path.join(_ROOT_PATH,
'django_extjs_tut.sqlite3'),                      # Or path to
database file if using sqlite3.
        'USER': '',                      # Not used with
sqlite3.
        'PASSWORD': '',                  # Not used with
sqlite3.
        'HOST': '',                      # Set to empty string
for localhost. Not used with sqlite3.
        'PORT': '',                      # Set to empty string
for default. Not used with sqlite3.
    }
}
```

- Adjust `STATICFILES` setting to

```
STATICFILES_DIRS = (
    os.path.join(_ROOT_PATH, 'static'), #project-wide static
files
    # Put strings here, like "/home/html/static" or
"C:/www/django/static".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
    )
```

  We don't add any paths to app-specific `static/` directories - they will be detected automatically by Django as long as there is `'django.contrib.staticfiles.finders.AppDirectoriesFinder'` in `STATICFILES_FINDERS`.

- Adjust `TEMPLATE_DIRS` setting to

```
TEMPLATE_DIRS = (
    os.path.join(_ROOT_PATH, 'templates'), #project-wide
templates

    #app templates/ dirs are detected automatically, so
they'are not put there

    # Put strings here, like "/home/html/django_templates" or
"C:/www/django/templates".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
)
```

  Again there is no need to specify app-specific `templates/` directories.

6. Now, test if everything worked

   ```
   python manage.py syncdb
   ```

   and possibly

   ```
   python manage.py runserver
   ```

7. C reate a file `base.html` in `django_extjs_tutorial/templates/` with a content:

   ```
   <!-- Do NOT put any DOCTYPE here unless you want problems in IEs. -
   ->
   <html>

   <head>
           <!-- The following line defines content type and utf-8 as
   character set. -->
           <!-- If you want your application to work flawlessly with
   various local -->
           <!-- characters, just make ALL strings, on the page, json
   and database utf-8. -->
           <meta http-equiv="Content-Type" content="text/html;
   charset=utf-8">

           <!-- Ext relies on its default css so include it here. -->
           <!-- This must come BEFORE javascript includes! -->
           <link rel="stylesheet" type="text/css" href="/static/ext-
   3.3.1/resources/css/ext-all.css">

           <!-- 3rd party css -->

           <!-- Include here your own css files if you have them. -->

           <!-- First of javascript includes must be an adapter... -->
           <script type="text/javascript" src="/static/ext-
   3.3.1/adapter/ext/ext-base.js"></script>

           <!-- ...then you need the Ext itself, either debug or
   production version. -->
           <script type="text/javascript" src="/static/ext-3.3.1/ext-
   all-debug.js"></script>


           <!-- Main js program -->


           <!-- Set a title for the page (id is not necessary). -->
           <title>{% block title %}My amazing site{% endblock %}
   </title>

           <!-- You can have onReady function here or in your
   application file. -->
           <!-- If you have it in your application file delete the
   whole -->
           <!-- following script tag as we must have only one onReady.
   ```

```
-->
        <script type="text/javascript">
        // Path to the blank image must point to a valid location
on your server
        Ext.BLANK_IMAGE_URL = '/static/ext-
3.3.1/resources/images/default/s.gif';
        </script>

        {% block head %}{% endblock %}

</head>

<body>
        {% block content %}{% endblock %}
</body>

</html>
```

This template contains all ExtJs includes we need for development. App templates will extend this template. Note: Change `ext-3.3.1` name in this template, it should correspond to ExtJs directory name in `django_extjs_tutorial/static/`.

PART 1. Create the hello_extjs app

This app will just use ExtJs as in http://www.sencha.com/learn/Tutorial:Getting_Productive#MessageBox.

1. Create an app

```
python manage.py startapp hello_extjs
```

2. Open `settings.py` and add `hello_extjs` to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'hello_extjs',
)
```

3.
```
python manage.py syncdb
```

4.
```
cd hello_extjs/
```

5. Create two new directories in `hello_extjs/`:
    - `static/` - this may store app-specific static files, like CSS, JavaScript.
    - `templates/` - this may store app-specific HTML templates.

    ```
    mkdir static
    mkdir templates
    ```

    Both directories have standard names, and they will be automatically found by Django as long as:
    - `'django.contrib.staticfiles.finders.AppDirectoriesFinder'` is in `STATICFILES_FINDERS`
    - `'django.template.loaders.app_directories.Loader'` is in `TEMPLATE_LOADERS` They are there by default 🙂
6. In both `hello_extjs/static/` and `hello_extjs/templates/` crate a subdirectory `hello_extjs` (full paths:`hello_extjs/static/hello_extjs` and `hello_extjs/templates/hello_extjs`).

    ```
    mkdir static/hello_extjs
    mkdir templates/hello_extjs
    ```

    Yet another subdirectory is necessary if we intend to add other apps that may have static files and templates with the same name - we will refer to file and template names using e.g. `app_name/template.html` (see more at Django Reusable App Conventions).Your directory now should look like this:

    ```
    django_extjs_tutorial/
       django_extjs_tut.sqlite3
       hello_extjs/
          __init__.py
          models.py
          static/
             hello_extjs/
          templates/
             hello_extjs/
          tests.py
          views.py
       __init__.py
       manage.py
       settings.py
       static/
          ext-3.3.1/
             ...
       templates/
          base.html
       urls.py
    ```
7. Open `django_extjs_tutorial/urls.py` and add

    ```
        (r'^hello_extjs/', include('hello_extjs.urls')),
    ```

to `urlpatterns`:

```
urlpatterns = patterns('',
    (r'^hello_extjs/', include('hello_extjs.urls')),
    # Example:
    # (r'^django_extjs_tutorial/',
include('django_extjs_tutorial.foo.urls')),

    # Uncomment the admin/doc line below to enable admin
documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    # (r'^admin/', include(admin.site.urls)),
)
```

8. Create a file `urls.py` in `hello_extjs/` (`hello_extjs/urls.py`) and fill with the following

```
from django.conf.urls.defaults import patterns, include

# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('hello_extjs.views',
    (r'^$', 'index'),
)
```

9. In `hello_extjs/templates/hello_extjs` create `index.html` looking like

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

{% block title %}Hello ExtJs{% endblock %}

{% block head %}
  <script type='text/javascript' src='/static/hello_extjs.js'>
</script>
{% endblock %}

{% block content %}

<p>Click on me</p>

{% endblock %}
```

10. In `hello_extjs/static/hello_extjs` create `hello_extjs.js` looking like

```
Ext.onReady(function() {
    var paragraphClicked = function(e) {
        var paragraph = Ext.get(e.target);
        paragraph.highlight();
        Ext.MessageBox.show({
            title: 'Paragraph Clicked',
            msg: paragraph.dom.innerHTML,
            width:400,
            buttons: Ext.MessageBox.OK,
            animEl: paragraph
        });
    }
    Ext.select('p').on('click', paragraphClicked);
});
```

11. In `hello_extjs/views.py` add a view

```
from django.shortcuts import render_to_response


def index(request):
    return render_to_response('hello_extjs/index.html')
```

12. Now in a web browser navigate to `http://localhost:8000/hello_extjs/`. You should see "Click on me" text. So click on it, and you should see a message box 🙂

PART 2. Using Ext.Direct

This app will just use ExtJs as in http://dev.sencha.com/deploy/dev/examples/direct/direct-tree.php. Download and install extdirect http://pypi.python.org/pypi/extdirect/

1. Create an app

```
python manage.py startapp direct_tree
```

2. Open `settings.py` and add the following apps to `INSTALLED_APPS`:
    - `direct_tree`
    - `extdirect.django`

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'hello_extjs',
    'extdirect.django',
    'direct_tree',
)
```

3.
```
python manage.py syncdb
```

4.
```
cd direct_tree/
```

5. Create two new directories in `direct_tree/`:
   - `static/direct_tree` - this may store app-specific static files, like CSS, JavaScript.
   - `templates/direct_tree` - this may store app-specific HTML templates.

```
mkdir static
mkdir static/direct_tree
mkdir templates
mkdir templates/direct_tree
```

6. Open `django_extjs_tutorial/urls.py` and add

```
import extdirect.django as extdirect
extdirect.autodiscover()
```

at the beginning of the file and

```
(r'^direct-tree/', include('direct_tree.urls')),
(r'^extdirect/', include('extdirect.django.urls')),
```

to `urlpatterns`:

```
import extdirect.django as extdirect
extdirect.autodiscover()

urlpatterns = patterns('',
    (r'^hello_extjs/', include('hello_extjs.urls')),
    (r'^direct-tree/', include('direct_tree.urls')),
    (r'^extdirect/', include('extdirect.django.urls')),
    # Example:
    # (r'^django_extjs_tutorial/',
include('django_extjs_tutorial.foo.urls')),

    # Uncomment the admin/doc line below to enable admin
documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    # (r'^admin/', include(admin.site.urls)),
)
```

7. Create a file `urls.py` in `direct_tree/` (`direct_tree/urls.py`) and fill with the following

```
from django.conf.urls.defaults import patterns, include
```

```
# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('direct_tree.views',
    (r'^$', 'index'),
)
```

8. In `direct_tree/templates/direct_tree` create `index.html` looking like

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



{% block title %}Ext Direct tree{% endblock %}
{% block head %}

{% load direct_providers %}
{% direct_providers %}

<script type='text/javascript'
src='/static/direct_tree/direct_tree.js'></script>
{% endblock %}

{% block content %}

{% endblock %}
```

Note

```
{% load direct_providers %}
{% direct_providers %}
```

lines in **header**. They will automatically load providers (no need to use `Ext.Direct.addProvider` manually in your scripts!!!)

9. In `direct_tree/static/direct_tree` create `direct_tree.js` looking like

```
Ext.onReady(function(){

        var tree = new Ext.tree.TreePanel({
                width: 400,
                height: 400,
                autoScroll: true,
                renderTo: document.body,
                root: {
                    id: 'root',
                    text: 'Root'
                },
                loader: new Ext.tree.TreeLoader({
                        paramsAsHash: true,
                        directFn: Remote.MyRouter.getTree,
                        nodeParameter: 'id'
                    }),
                fbar: [{
                        text: 'Reload root',
                        handler: function(){
                            tree.getRootNode().reload();
                        }
                    }]
            });
    });
```

Note, that this code is a little bit different than that on ExtJS examples. First, we don't used `Ext.Direct.addProvider` as these are added automatically by `extdirect.django`. Second, we use `paramsAsHash` parameter in `TreeLoader` - as our django code expects the request formatted as key/value pairs. Third, we use `nodeParameter` set to `'id'`. The default nodeParameter is 'node', which is perfectly fine, but we changed it to make our Django server code more similar to PHP code from the original Direct Tree example

10. In `direct_tree/views.py` add a view

```
from django.shortcuts import render_to_response


def index(request):
    return render_to_response('direct_tree/index.html')
```

11. Create a file `direct_tree/direct.py` with:

```
from extdirect.django import DirectRouter, register_router

class MyRouter(DirectRouter):
    def getTree(self, id):
        out = []
        if id == 'root':
            for i in range(1, 6):
                out.append(
                           {
                            'id': 'n' + str(i),
                            'text': 'Node ' + str(i),
                            'leaf': False,
                            }
                           )

        elif len(id)==2:
            num = id[1]
            for i in range(1, 6):
                out.append(
                           {
                            'id': id + str(i),
                            'text': 'Node ' + num + '.' + str(i),
                            'leaf': True,
                            }
                           )
        return out

register_router(MyRouter, 'Remote')
```

This code will be executed automatically on startup. Alternatively, this can be added to `views.py` or file with different name. Then, it is necessary to add `import views` or `import that_different_filename` in `direct_tree/__init__.py` to run the code on startup.

Note that `getTree` function returns python list, not json string - extdirect module will convert it to json automatically.

12. N ow **run or restart** the server and in a web browser navigate to `http://localhost:8000/direct-tree/`.