URL Namespacing and Views

This reading explains the use of named URLs in the **URLconf** of a Django app and how the use of a namespace helps in resolving the same URL name in more than one app.

In each app, there is a **urls.py** file that defines the list of URL patterns for that app. Each pattern is constructed by **django.urls.path()** function. Its arguments are a URL path string, the name of the view function to be mapped to it and an optional argument name.

The following is the **urls.py** of an app:

```
#demoapp/urls.py
from django.urls import path
from . import views

 urlpatterns = [
   path('', views.index, name='index'),
   path('login/', views.login, name='login')
]
```

It is included in the project's **urlpatterns**.

```
from django.contrib import admin
from django.urls import path, include

 urlpatterns = [
   path('admin/', admin.site.urls),
   path('demo/', include('demoapp.urls')),
]
```

Normally, the client's request URL is mapped against the function so that the application flow is directed toward it.

The URL references used internally by the application are hard-coded strings.

For example, a form template's action attribute points towards **/demoapp/login** URL so that when the form is submitted, the login view mapped to this URL is invoked.

```
<form action='/demoapp/login', method='POST>
    # ...
</form>
```

reverse() function

However, the hard-coded URLs make the application less scalable and difficult to maintain as the project grows. In such a case, you can obtain the URL from the name parameter used in the **path()** function.

Start the Django shell.

```
python manage.py shell
```

Django's **reverse()** function returns the URL path to which it is mapped.

```
>>> from django.urls import reverse
>>> reverse('index')
'/demo/'
```

The problem comes when the view function of the same name is defined in more than one app. This is where the idea of a **namespace** is needed.

Application namespace
The application namespace is created by defining **app_name** variable in the application's **urls.py** and assigning it the name of the app. In the **demoapp/urls.py** script, make the change using the following code:

```python
#demoapp/urls.py
from django.urls import path
from . import views
app_name='demoapp'
urlpatterns = [
    path('', views.index, name='index'),
]
```

The **app_name** defines the application namespace so that the views in this app are identified by it.

To obtain the URL path of the **index()** function, call the **reverse()** function by prepending the namespace to it.

```
>>> reverse('demoapp:index')
'/demo/'
```

To appreciate the advantage of defining a namespace, add another app in the project, for example, **newapp**. Provide an **index()** view function in it and define **app_name** in its **URLConf** file (that is **urls.py**).

```python
#newapp/urls.py
from django.urls import path
from . import views
app_name='newapp'
urlpatterns = [
    path('', views.index, name='index'),
]
```

Update the project's **urls.py**.

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('demo/', include('demoapp.urls')),
    path('newdemo/', include('newapp.urls')),
]
```

The **reverse()** function is executed for index view in **newapp** namespace.

```
>>> reverse('newapp:index')
'/newdemo/'
```

You can see that Django differentiates between same-named URLs in multiple apps with application namespace.

Instance namespace
You can also use the namespace parameter in the **include()** function while adding an app's **urlpattern** to that of a project.

```python
#in demoproject/urls.py
urlpatterns=[
    # ...
    path('demo/', include('demoapp.urls', namespace='demoapp'))
    path('demo/', include('demoapp.urls', namespace='demoapp')),
    # ...
]
```

This namespace is called the **`instance namespace`**.

Using namespace in view
Suppose you want the user to be conditionally redirected to the login view from inside another view. You need to obtain the URL of the login view and send the control to it with **`HttpResponsePermanentRedirect`**.

```
from django.http import HttpResponsePermanentRedirect
from django.urls import reverse

def myview(request):
    ....
    return HttpResponsePermanentRedirect(reverse('demoapp:login'))
```

namespace in the **`url`** tag
An HTML form is submitted to the URL specified in the action attribute.

```
<form action="/demoapp/login" method="post">

#form attributes

<input type='submit'>

</form>
```

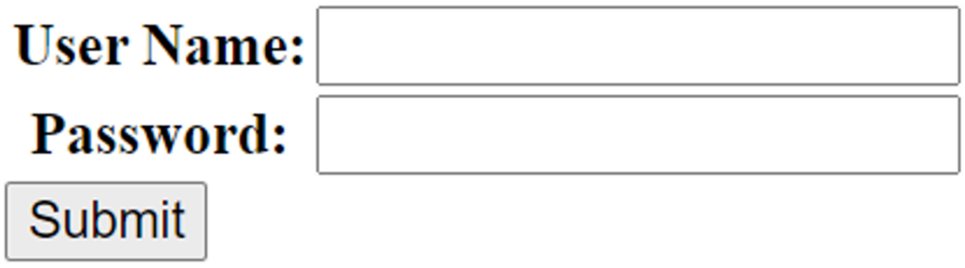
The form will then be processed by the view mapped to this URL. However, as mentioned above, a hard-coded URL is not desired. Instead, use the **`url`** tag of the Django Template Language. It returns the absolute path from the named URL.
Use the **`url`** tag to obtain the URL path dynamically, as shown below:

```
<form action="{% url 'login' %}" method="post">
#form attributes
<input type='submit'>
</form>
```

Again, the login view may be present in multiple apps. Use the named URL qualified with the namespace to resolve the conflict.

```
<form action="{% url 'demoapp:login' %}" method="post">
#form attributes
<input type='submit'>
</form>
```

The browser shows the login form as below:



Thus, the concept of namespace helps in resolving the conflict arising out of multiple apps under the same project having views of the same name.
You have covered some of the concepts here in line with the use of form templates in Django. If you have difficulty understanding some things, be assured it will be much more clear once you cover the topic of templates in this course.
In this reading, you further explored the concept of URL Namespacing and Views in Django.