In Chapter 3, we explained the basics of Django view functions and URLconfs. This chapter goes into more detail about advanced functionality in those two pieces of the framework.

**URLconf Tricks**

There's nothing "special" about URLconfs—like anything else in Django, they're just Python code. You can take advantage of this in several ways, as described in the sections that follow.

**Streamlining Function Imports**

Consider this URLconf, which builds on the example in Chapter 3:

```python
from django.conf.urls.defaults import *
from mysite.views import current_datetime, hours_ahead, hours_behind,
now_in_chicago, now_in_london

urlpatterns = patterns(''
    (r'^now/$', current_datetime),
    (r'^now/plus\d{1,2}hours/$', hours_ahead),
    (r'^now/minus\d{1,2}hours/$', hours_behind),
    (r'^now/in_chicago/$', now_in_chicago),
    (r'^now/in_london/$', now_in_london),
)
```

As explained in Chapter 3, each entry in the URLconf includes its associated view function, passed directly as a function object. This means it's necessary to import the view functions at the top of the module.

But as a Django application grows in complexity, its URLconf grows, too, and keeping those imports can be tedious to manage. (For each new view function, you have to remember to import it, and the import statement tends to get overly long if you use this approach.) It's possible to avoid that tedium by importing the views module itself. This example URLconf is equivalent to the previous one:

```python
from django.conf.urls.defaults import *
from mysite import views
```
urlpatterns = patterns(''
    (r'^now/$', views.current_datetime),
    (r'^now/plus\(\d\{1,2\}\)hours/$', views.hours_ahead),
    (r'^now/minus\(\d\{1,2\}\)hours/$', views.hours_behind),
    (r'^now/in_chicago/$', views.now_in_chicago),
    (r'^now/in_london/$', views.now_in_london),
)'
)

Django offers another way of specifying the view function for a particular pattern in the URLconf: you can pass a string containing the module name and function name rather than the function object itself. Continuing the ongoing example:

from django.conf.urls.defaults import *

urlpatterns = patterns(''
    (r'^now/$', 'mysite.views.current_datetime'),
    (r'^now/plus\(\d\{1,2\}\)hours/$', 'mysite.views.hours_ahead'),
    (r'^now/minus\(\d\{1,2\}\)hours/$', 'mysite.views.hours_behind'),
    (r'^now/in_chicago/$', 'mysite.views.now_in_chicago'),
    (r'^now/in_london/$', 'mysite.views.now_in_london'),
)'
)

(Note the quotes around the view names. We're using 'mysite.views.current_datetime'—with quotes—instead of mysite.views.current_datetime.)

Using this technique, it's no longer necessary to import the view functions. Django automatically imports the appropriate view function the first time it's needed, according to the string describing the name and path of the view function.

A further shortcut you can take when using the string technique is to factor out a common "view prefix." In our URLconf example, each of the view strings starts with 'mysite.views', which is redundant to type. We can factor out that common prefix and pass it as the first argument to patterns(), like this:

from django.conf.urls.defaults import *

urlpatterns = patterns('mysite.views',
    (r'^now/$', 'current_datetime'),
    (r'^now/plus\(\d\{1,2\}\)hours/$', 'hours_ahead'),
    (r'^now/minus\(\d\{1,2\}\)hours/$', 'hours_behind'),
    (r'^now/in_chicago/$', 'now_in_chicago'),
    (r'^now/in_london/$', 'now_in_london'),
)

(Note that you don't put a trailing dot (".")) in the prefix, nor do you put a leading dot in the view strings. Django puts those in automatically.

With these two approaches in mind, which is better? It really depends on your personal coding style and needs.

Advantages of the string approach are as follows:

- It's more compact, because it doesn't require you to import the view functions.
- It results in more readable and manageable URLconfs if your view functions are spread across several different Python modules.