Django - URL Mapping

Now that we have a working view as explained in the previous chapters. We want to access that view via a URL. Django has his own way for URL mapping and it's done by editing your project url.py file **(myproject/url.py)**. The url.py file looks like −

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
   #Examples
   #url(r'^$', 'myproject.view.home', name = 'home'),
   #url(r'^blog/', include('blog.urls')),

   url(r'^admin', include(admin.site.urls)),
)
```

When a user makes a request for a page on your web app, Django controller takes over to look for the corresponding view via the url.py file, and then return the HTML response or a 404 not found error, if not found. In url.py, the most important thing is the **"urlpatterns"** tuple. It’s where you define the mapping between URLs and views. A mapping is a tuple in URL patterns like −

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
   #Examples
   #url(r'^$', 'myproject.view.home', name = 'home'),
   #url(r'^blog/', include('blog.urls')),

   url(r'^admin', include(admin.site.urls)),
   url(r'^hello/', 'myapp.views.hello', name = 'hello'),
)
```

The marked line maps the URL "/home" to the hello view created in myapp/view.py file. As you can see above a mapping is composed of three elements −

The pattern − A regexp matching the URL you want to be resolved and map. Everything that can work with the python 're' module is eligible for the pattern (useful when you want to pass parameters via url).

The python path to the view − Same as when you are importing a module.
The name − In order to perform URL reversing, you'll need to use named URL patterns as done in the examples above. Once done, just start the server to access your view via :http://127.0.0.1/hello

Organizing Your URLs

So far, we have created the URLs in "myprojects/url.py" file, however as stated earlier about Django and creating an app, the best point was to be able to reuse applications in different projects. You can easily see what the problem is, if you are saving all your URLs in the "projecturl.py" file. So best practice is to create an "url.py" per application and to include it in our main projects url.py file (we included admin URLs for admin interface before).

Organize URLs
Image not found or type unknown

How is it Done?

We need to create an url.py file in myapp using the following code −

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

urlpatterns = patterns('', url(r'^hello/', 'myapp.views.hello', name = 'hello'),)
```

Then myproject/url.py will change to the following −

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
   #Examples
   #url(r'^$', 'myproject.view.home', name = 'home'),
   #url(r'^blog/', include('blog.urls')),

   url(r'^admin', include(admin.site.urls)),
   url(r'^myapp/', include('myapp.urls')),
)
```

We have included all URLs from myapp application. The home.html that was accessed through "/hello" is now "/myapp/hello" which is a better and more understandable structure for the web app.

Myproject
Image not found or type unknown

Now let's imagine we have another view in myapp “morning” and we want to map it in myapp/url.py, we will then change our myapp/url.py to −

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

urlpatterns = patterns('',
   url(r'^hello/', 'myapp.views.hello', name = 'hello'),
   url(r'^morning/', 'myapp.views.morning', name = 'morning'),
)
```

This can be re-factored to −

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

urlpatterns = patterns('myapp.views',
   url(r'^hello/', 'hello', name = 'hello'),
   url(r'^morning/', 'morning', name = 'morning'),)
```

As you can see, we now use the first element of our **urlpatterns** tuple. This can be useful when you want to change your app name.

URL Patterns
Image not found or type unknown

Sending Parameters to Views

We now know how to map URL, how to organize them, now let us see how to send parameters to views. A classic sample is the article example (you want to access an article via “/articles/article_id”).

Passing parameters is done by capturing them with the **regexp** in the URL pattern. If we have a view like the following one in “myapp/view.py”

```
from django.shortcuts import render
from django.http import HttpResponse

def hello(request):
   return render(request, "hello.html", {})

def viewArticle(request, articleId):
   text = "Displaying article Number : %s"%articleId
   return HttpResponse(text)
```

We want to map it in myapp/url.py so we can access it via “/myapp/article/articleId”, we need the following in “myapp/url.py” −

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

urlpatterns = patterns('myapp.views',
   url(r'^hello/', 'hello', name = 'hello'),
   url(r'^morning/', 'morning', name = 'morning'),
   url(r'^article/(\d+)/', 'viewArticle', name = 'article'),)
```

When Django will see the url: “/myapp/article/42” it will pass the parameters '42' to the viewArticle view, and in your browser you should get the following result −

Passing parameters to viewArticle

Note that the order of parameters is important here. Suppose we want the list of articles of a month of a year, let's add a viewArticles view. Our view.py becomes −

```
from django.shortcuts import render
from django.http import HttpResponse

def hello(request):
   return render(request, "hello.html", {})

def viewArticle(request, articleId):
   text = "Displaying article Number : %s"%articleId
   return HttpResponse(text)

def viewArticle(request, month, year):
   text = "Displaying articles of : %s/%s"%(year, month)
   return HttpResponse(text)
```

The corresponding **url.py** file will look like −

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

urlpatterns = patterns('myapp.views',
   url(r'^hello/', 'hello', name = 'hello'),
   url(r'^morning/', 'morning', name = 'morning'),
   url(r'^article/(\d+)/', 'viewArticle', name = 'article'),
   url(r'^articles/(\d{2})/(\d{4})', 'viewArticles', name = 'articles'),)
```

Now when you go to “/myapp/articles/12/2006/” you will get 'Displaying articles of: 2006/12' but if you reverse the parameters you won’t get the same result.

Displaying Articles

To avoid that, it is possible to link a URL parameter to the view parameter. For that, our **url.py** will become −

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

urlpatterns = patterns('myapp.views',
   url(r'^hello/', 'hello', name = 'hello'),
   url(r'^morning/', 'morning', name = 'morning'),
   url(r'^article/(\d+)/', 'viewArticle', name = 'article'),
   url(r'^articles/(?P\d{2})/(?P\d{4})', 'viewArticles', name = 'articles'),)
```

Revision #1
Created 14 December 2022 11:06:24 by Admin
Updated 14 December 2022 11:06:55 by Admin