# 7. Les URL de Django

Nous sommes sur le point de créer notre première page web : une page d'accueil pour votre blog ! Mais d'abord, commençons par en apprendre un peu plus sur les URL de Django.

Qu'est-ce qu'une URL ?

Une URL est une adresse web. Vous pouvez voir une URL chaque fois que vous visitez un site web. Elle est visible dans la barre d'adresse de votre navigateur. (Oui ! 127.0.0.1:8000 est aussi une URL ! Et https://djangogirls.org est aussi une URL.)

Chaque page internet a besoin de sa propre URL. C'est grâce à cela que votre application sait ce qu'elle doit afficher à un utilisateur qui ouvre cette URL. Dans Django, nous utilisons un outil appelé "URLconf" (Configuration d'URL). URLconf est un ensemble de modèles que Django va essayer de faire correspondre à l'URL demandée afin de trouver la vue adaptée.

Comment les URLs fonctionnent-elles dans Django ?

Ouvrons le fichier mysite/urls.py dans notre éditeur de code et regardons à quoi il ressemble :

`mysite/urls.py`

```
"""
mysite URL Configuration

[...]
"""
from django.contrib import admin
from django.urls import path

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

Comme vous pouvez le voir ici, Django a déjà préparé quelque chose pour nous.

Les lignes entre triples guillemets (' ou """) sont appelées docstrings. Vous pouvez les écrire en haut d'un fichier, d'une classe ou d'une méthode pour décrire son fonctionnement. Elle ne seront pas exécutées par Python.

L'URL admin, que vous avez abordé dans le chapitre précédent est déjà là :

`mysite/urls.py`

```
  path('admin/', admin.site.urls),
```

Cette ligne signifie que pour toutes les URL commençant par "admin/", Django trouvera une vue correspondante. Dans cet exemple, nous incluons de nombreuses URL admin. Elles ne sont donc pas toutes contenues dans ce petit fichier. C'est plus lisible et plus propre.

Votre première URL Django !

Il est temps de créer notre première URL ! Nous voulons que 'http://127.0.0.1:8000/' soit la page d'accueil de notre blog et qu'elle affiche la liste des articles.

Nous aimerions aussi garder notre fichier mysite/urls.py propre. Pour cela, nous allons importer les URLs de notre application blog dans notre fichier principal mysite/urls.py.

Allez-y ! Ajoutez une ligne qui importera blog.urls. Vous devrez également modifier la ligne "from django.urls..." car nous utiliseront la fonction "include". Il vous faudra ajouter cet import à la ligne.

Votre fichier mysite/urls.py devrait maintenant ressembler à ceci :

`mysite/urls.py`

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

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

Django va maintenant rediriger tout ce qui arrive sur "http://127.0.0.1:8000/" vers blog.urls puis regardera dans ce fichier pour y trouver la suite des instructions à suivre.

blog.urls

Créez un nouveau fichier vide nommé urls.py dans le répertoire blog, et ouvrez-le dans l'éditeur de code. Très bien ! Ajoutez ces deux premières lignes :

`blog/urls.py`

```
from django.urls import path
from . import views
```

Nous venons d'importer la fonction path de Django ainsi que toutes les vues liées à notre application blog. (Cependant, nous n'avons pas encore créé de vues ! Pas de problème : nous y viendrons dans une minute !)

Après ça, nous pouvons ajouter notre premier modèle (pattern) d'URL:

`blog/urls.py`

```
urlpatterns = [
    path('', views.post_list, name='post_list'),
]
```

Comme vous pouvez le voir, nous assignons une vue appelée post_list à l'URL racine. Ce modèle d'URL correspond à une chaîne vide et le résolveur d'URL de Django ignore le nom de domaine (par exemple, http://127.0.0.1:8000/), soit la première partie de l'URL. Ce pattern va donc indiquer à Django d'afficher la vue views.post_list à un utilisateur de votre site web qui se rendrait à l'adresse "http://127.0.0.1:8000/".

La dernière partie, name='post_list', est le nom de l'URL qui sera utilisée afin d'identifier la vue. Ce nom peut être le même que celui de la vue ou quelque chose de complètement différent. Plus tard dans ce tutoriel, nous allons utiliser les noms que nous avons donné à nos URLs. Il est donc important de donner un nom unique à chaque URL que nous créons. Pour vous faciliter la tâche, essayez de trouver des noms d'URLs simple à retenir.

Si vous essayez d'aller sur "http://127.0.0.1:8000/" maintenant, vous trouverez un message du style "page web non disponible". C'est parce que le serveur (vous vous souvenez d'avoir tapé runserver ?) n'est plus en exécution. Jetez un œil à la console pour savoir pourquoi.

Votre console affiche une erreur, mais ne vous inquiétez pas – c'est en fait très utile : elle vous dit qu'il n'existe aucun attribut "post_list" (no attribute 'post_list'). C'est le nom de la vue que Django essaie de trouver et d'utiliser, or nous ne l'avons pas encore créée. À ce stade, votre /admin/ ne fonctionnera pas non plus. Pas de problème, on y vient. Si vous voyez un message d'erreur différent, essayez de redémarrer votre serveur web. Pour faire cela, dans la console qui exécute le serveur web, arrêtez-le en appuyant sur CTRL+C (les touches Control et C simultanément). Sous Windows, vous devrez peut-être appuyer sur Ctrl+Break. Vous devrez alors redémarrer le serveur web en exécutant la commande "python manage.py runserver".

8. Les vues