# Développement web côté serveur

# Framework python-django

# Travaux Pratiques N°5

# (Projet Mysite : Suite TP N°3)

# Configuration et héritage

# des Templates

**Objectifs du TP:**

* **Configuration des Templates**
* **Fichiers Statiques**
* **Template et Fichiers Statiques**
* **Héritage des Templates**

Les templates sont, à la base, des fichiers HTML affichés par votre navigateur. Tous les sites Web, qu'il s'agisse de sites Web simples et statiques ou d'applications Web interactives fonctionnant sur plusieurs appareils, sont basés sur du code HTML, du code CSS et des scripts en langage JavaScript qui s’exécutent en front end. Le back end peut utiliser le framework python-django pour faire le traitement métier et chercher les données dans le système d’information pour les afficher sur des templates.

L'approche de Django en matière de conception de sites Web est simple: laissez la logique et le code de Django séparés du design. Cela signifie qu'il est possible pour un concepteur de créer un frontal complet (HTML, CSS, imagerie et interaction utilisateur) sans jamais avoir à écrire une seule ligne de code Python ou Django.

Au cours de ce TP, vous apprendrez à créer un template HTML simple, puis à ajouter des balises de Template Django aux fichiers HTML de base pour créer un Template capable d'afficher les données de la base de données.

Avant de faire cela, nous devons replonger dans les paramètres et la structure de votre projet afin que vous puissiez comprendre où Django recherche les templates et comment il décide quel template afficher.

**Configurations des Templates**

Pour que Django puisse afficher votre template de site, il doit d'abord savoir où chercher le (s) fichier (s) templates. Ceci est réalisé par le paramètre **TEMPLATES** dans le fichier settings.py. La liste ci-dessous montre une liste partielle des paramètres du template:

Les lignes importantes dans cette liste de paramètres sont les lignes :

* **DIRS**: contient une liste de chemins d'accès aux dossiers contenant des templates. Les chemins peuvent être absolus ou relatifs. La valeur par défaut est une liste vide.
* **APP\_DIRS : True,** Django recherchera un dossier nommé **templates** dans chacune de vos applications. La valeur par défaut est True.

**Fichier settings.py:**

Diagram

Description automatically generated

…………………………………………..

Une image contenant texte

Description générée automatiquement

Tous les fichiers templates ne seront pas liés à une application particulière. Le paramètre DIRS est utile pour créer des liens vers des templates qui existent ailleurs que dans votre structure de projet.

Dans notre projet, nous aurons un template de site qui ne fait pas partie de l'application pages, nous devons donc ajouter un chemin vers DIRS comme indiqué ci-haut.

N'oubliez pas d'importer le module os.

La commande semble compliquée, mais est facile à comprendre —os.path.join est une commande Python permettant de créer un chemin de fichier en joignant des chaînes (concaténation). Dans cet exemple, nous joignons mysite / templates au répértoire de notre projet pour créer le chemin complet vers notre répertoire de templates, c'est-à-dire, **E:/mydjango/mysite/mysite/templates**.

**Noms de chemins de fichiers dans Django**

Django utilise des chemins de style Linux dans de nombreux endroits: le fichier **settings.py** en fait partie.

Cela peut être un piège pour les programmeurs Windows. Assurez-vous d'utiliser la barre oblique (**/**) et pas la barre oblique inverse (**\**) dans les chemins d'accès au fichier de paramètres.

Cette règle s'applique également aux chemins de templates dans les vues Django que nous verrons plus loin dans ce TP.

Avant de passer à autre chose, assurez vous que vous avez créé un répertoire de templates dans le répertoire mysite interne comme le montre la figure ci-dessous.

Une image contenant texte

Description générée automatiquement

**Les Fichiers Static**

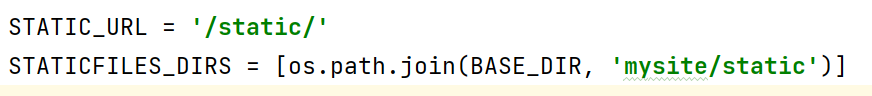
Django traite les fichiers statiques (images, CSS et JavaScript) différemment des templates. Les créateurs de Django voulaient qu’il soit rapide et évolutif, dès le début, Django a été conçu pour faciliter l’accès aux médias statiques stockés dans un serveur différent de celui sur lequel s'exécutait l'application Django principale.

Django améliore la vitesse et l'évolutivité en gardant les médias statiques dans un répertoire différent du reste de l'application. Ce répertoire est défini dans le fichier **settings.py** et est appelé **static** par défaut:

|  |
| --- |
| **STATIC\_URL = '/static/'** |

Cette ligne doit se trouver à la fin (ou proche de la fin) de votre fichier settings.py. Nous devons ajouter un autre paramètre afin que Django puisse trouver les fichiers statiques pour notre site.

Ajoutez, comme, ci-dessous le paramètre STATIC\_URL:



La liste **STATICFILES\_DIRS** joue le même rôle pour les fichiers statiques que celui joué par la liste **DIRS** pour les templates. Dans ce cas, nous demandons à Django de rechercher des fichiers statiques dans le répertoire **static** qui se trouve dans le dossier mysite interne. Maintenant, nous devons créer un dossier static dans le dossier mysite interne. Une fois le nouveau répertoire créé, votre répertoire de projet ressemblera à ceci:

Une image contenant texte

Description générée automatiquement

**Templates et Fichiers Statiques du site**

Maintenant que nous avons configuré Django pour pouvoir servir nos templates, il est temps de créer notre template de site et nos fichiers statiques.

Nous allons créer quatre fichiers:

1. **base.html**. Le template HTML de base pour notre site.

2. **main.css**. Les styles CSS pour notre template

3. **logo.jpg**. Une image pour le logo.

4. **top\_banner.png**. Une bannière pour le site

Les listings des fichiers **base.html** et main.css sont fournis ci-dessous. Le commentaire en haut de chaque liste vous montre où créer le fichier.

|  |
| --- |
| base.html |
| # \mysite\mysite\templates\base.html  **1** {% load static %}  **2** <!doctype html>  **3** <html>  **4** <head>  **5** <meta charset="utf-8">  **6** <title>Untitled Document</title>  **7** <link href="{% static 'main.css' %}" rel="stylesheet" type="text/css">  **8** </head>  **9** <body>  **10** <div id="wrapper">  **11** <header id="header">  **12** <div id="logo"><img src="{% static 'logo.jpg' %}" alt=""/></div>  **13** <div id="topbanner"><img src="{% static 'estc.jpg' %}"   width="800" height="200" alt=""/></div>  **14** </header>  **15** <aside id="leftsidebar">  **16** <nav id="nav">  **17** <ul>  <li>Menu 1</li>  <li>Menu 2</li>  <li>Menu 3</li>  </ul>  **18** </nav>  **19** </aside>  **20** <section id="main">  **21** <h1>Welcome!</h1>  **22** <p>This is the site template</p>  **23** </section>  **24** <footer id="footer">Copyright &copy; 2018 LPGLAASRI WebDesign</footer>  **25** </div>  **26** </body>  **27** </html> |

Ce fichier est principalement un fichier HTML5 normal. Noter les éléments sémantiques comme, <aside>, <section> et <footer>. Les éléments sémantiques fournissent une signification supplémentaire au navigateur sur la manière dont un élément de contenu doit être traité.

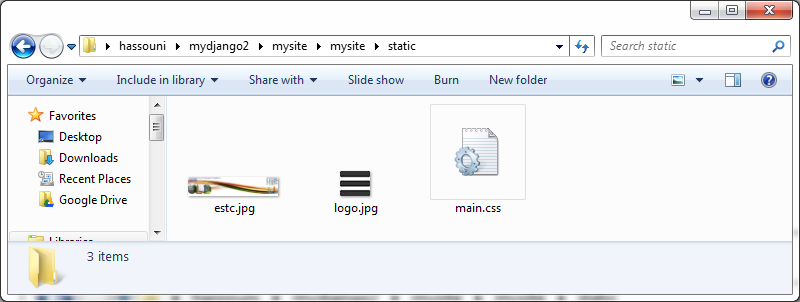
Le template base.html inclut également votre premier tag de template Django—**{% static %}.** La balise **static** est utilisée pour lier des médias dans vos templates au **STATIC\_ROOT** de votre projet. Comme nous sommes en développement et que nous n'avons pas configuré Django pour la production, **STATIC\_ROOT** est le même que le paramètre **STATIC\_URL** **(/static/**).

* **Ligne 1**. En premier, Nous chargeons la balise static dans notre template;
* **Lignes 7, 12 et 13**. Ensuite, partout où nous avons besoin de charger un média statique, nous transmettons le nom de fichier multimédia (par exemple, logo.jpg) à la balise **static**, ce qui ajoute automatiquement le répertoire statique static (par exemple, /static/logo.jpg).

|  |
| --- |
| main.css |
| # E:/mydjango/mysite/mysite/static/main.css  **1** @charset "utf-8";  **2** #header {  **3** border-style: none;  **4** width: 800px;  **5** height: auto;  **6** }  **7** #wrapper {  **8** margin-top: 0px;  **9** margin-left: auto;  **10** margin-right: auto;  **11** background-color: #FFFFFF;  **12** width: 800px;  **13** }  **14** body {  **15** background-color: #E0E0E0;  **16** font-family: Gotham, "Helvetica Neue", Helvetica,Arial, sans-serif;  **17** font-size: 0.9em;  **18** text-align: justify;  **19** color: #474747;  **20** }  **21** #footer {  **22** text-align: center;  **23** font-size: 0.8em;  **24** margin-top: 5px;  **25** padding-top: 10px;  **26** padding-bottom: 10px;  **27** background-color: #FFFFFF;  **28** border-top: thin solid #BBBBBB;  **29** clear: both;  **30** color: #969696;  **31** }  **32** #nav li {  **33** padding-top: 10px;  **34** padding-bottom: 10px;  **35** font-size: 1em;  **36** list-style-type: none;  **37** border-bottom: thin solid #5F5F5F;  **38** color: #4C4C4C;  **39** left: 0px;  **40** list-style-position: inside;  **41** margin-left: -10px;  **42** }  **43** #nav li a {  **44** text-decoration: none;  **45** }  **46** #leftsidebar {  **47** width: 180px;  **48** height: 350px;  **49** float: left;  **50** }  **51** #main {  **52** width: 560px;  **53** float: left;  **54** margin-left: 20px;  **55** margin-right: 10px;  **56** padding-right: 10px;  **57** }  **58** #logo {  **59** padding: 10px;  **60** } |

**logo.jpg and estc.jpg**

Ces fichiers doivent tous deux être placés dans le répertoire E:/mydjango/mysite/**mysite/static/**.



**Mise à jour de votre View**

Maintenant nous avons les templates et les fichiers statiques en place, nous devons mettre à jour notre **views.py**:

Une image contenant texte

Description générée automatiquement

|  |
| --- |
| # pages\views.py  **1** from django.shortcuts import render  **2** # from django.http import HttpResponse  **3**  **4** def index\_projet(request):  **5** # return HttpResponse("<h1>Bienvenue au site de GLAASRI</h1>")  **6 return render(request, "base.html" )** |

Pour notre nouvelle vue, nous avons remplacé l'appel à HttpResponse() par un appel à render().

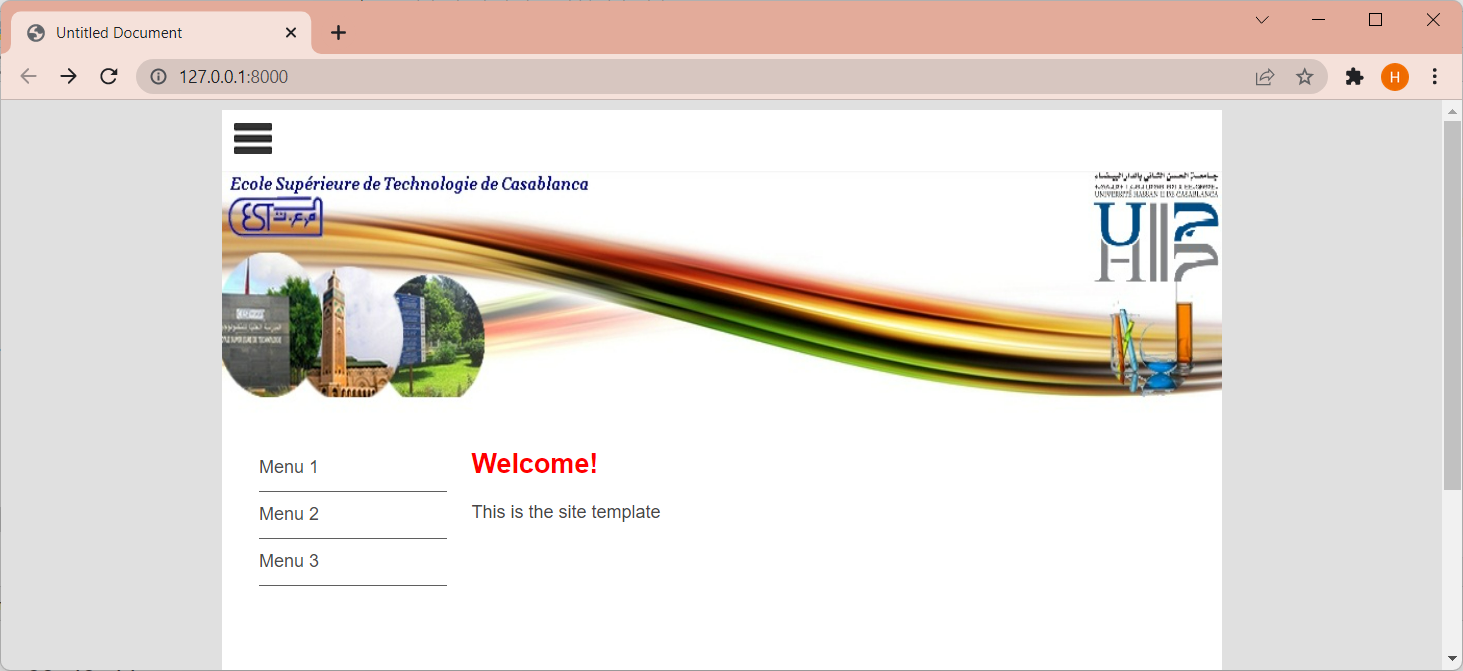
**render()** est une fonction d'assistance spéciale de Django qui crée un raccourci pour communiquer avec un navigateur Web. Django reçoit une requête d'un navigateur, il trouve la bonne vue et la vue renvoie une réponse au navigateur.

Dans un TP précédent, nous avons simplement renvoyé du texte HTML; Cependant, lorsque nous souhaitons utiliser un template Django doit d'abord charger le template, créer un contexte (qui est essentiellement un dictionnaire de variables et de données associées qui est renvoyé au navigateur), puis renvoyer une réponse HttpResponse.

**render()** fournit une méthode de raccourci dans laquelle vous fournissez la requête d'origine(l’objet **request**), le template et le contexte directement à la méthode render(). render() renvoie la réponse formatée de manière appropriée sans avoir à coder les étapes intermédiaires.

Dans notre **views.py** modifié, nous retournons simplement l'objet de requête d'origine du navigateur et le nom de notre template de site. Nous aborderons le contexte un peu plus tard dans ce TP.

Une fois que vous avez modifié votre fichier **views.py**, enregistrez-le et lancez le serveur de développement. Si vous accédez à **http://127.0.0.1:8000/**, vous devriez voir votre nouveau template de site.



**Le Template de l’application Pages**

Maintenant que le template de site est opérationnel, nous devons créer un template pour notre application **pages**. En appliquant le principe DRY (Don’t Repeat Your Self), nous ne voulons pas répéter la même information dans tous nos templates, nous voulons donc que notre template de l’application **pages** hérite du template de site.

L'implémentation de l'héritage est simple dans Django: vous définissez des blocs remplaçables dans chaque template parent afin que les templates enfants puissent remplacer des sections du template parent par du contenu propre à l'enfant.

C'est plus facile à comprendre avec un exemple, alors modifions **base.html** (les changements sont en gras):

|  |
| --- |
| # \mysite\mysite\templates\base.html  # ...  <section id="main">  **{% block content %}**  <h1 style=”color: blue;”>>Welcome!</h1>  <p>This is the site template</p>  **{% endblock content %}**  </section>  # ... |

Les deux lignes de code que nous avons ajoutées sont un ensemble de balises de bloc Django. Une balise de bloc définit un bloc de code de template parent qui peut être remplacé par tout template enfant qui hérite du template. Les tags de bloc ont la forme suivante:

**{% block <name> %}{% endblock <name> %}**

La deuxième déclaration **<name>** dans endeblock n'est pas obligatoire bien qu'elle soit fortement recommandée, en particulier si vous utilisez plusieurs balises de bloc. Vous pouvez nommer vos tags de bloc comme vous le souhaitez - dans notre exemple, nous nommons le tag de bloc par **content** .

Ensuite, nous devons créer un template pour notre application **pages** qui hérite du template de site. Si vous vous en souvenez plus tôt dans le TP, le paramètre **APP\_DIRS** est défini par défaut sur True. Cela signifie que Django recherchera dans toutes vos applications un dossier nommé templates. Créez alors un sous dossier templates dans le dossier de l'application pages. Nous devons ensuite créer un autre dossier à l'intérieur. Ce deuxième dossier est nommé pages qui est le nom de l'application. La structure de votre dossier d'application pages devrait ressembler à ceci lorsque vous avez terminé:

|  |
| --- |
| \pages  \templates  \pages |

Pourquoi un deuxième dossier **pages**?

Que faire si vous avez deux applications dans votre projet qui ont chacune un template nommé index.html? Django utilise une logique de court-circuit lors de la recherche de templates, alors quand il va chercher templates/index.html, il utilisera la première instance qu'il trouve et qui peut ne pas être dans l'application que vous vouliez!

L'ajout du dossier **pages** intérieure est un exemple **d'espace de noms** dans vos templates. En ajoutant ce dossier, vous pouvez vous assurer que Django récupère le bon template.

Créeons alors le template—**page.html**:

|  |
| --- |
| # \pages\templates\pages\page.html  **1** {% extends "base.html" %}  **2**  **3** {% block content %}  **4** <h1 style=”color: red;”>Welcome to GLAASRI Site !</h1>  **5** <p>This is the page template</p>  **6** {% endblock content %} |

Regardons de plus près ceci:

**Ligne 1**. C'est là où se manifeste l'intérêt de l'héritage. Nous disons à Django que le template **page.html** étend (hérite) le template de base (de site).

**Lignes 3 à 5**. Nous déclarons un ensemble de balises de bloc nommé content. Ce bloc remplacera le bloc du même nom dans le template parent.

Notez que nous n'avons pas répété une seule ligne de code depuis **base.html**: nous chargeons le fichier base.html avec la balise extends et remplaçons le bloc content par un nouveau bloc content du **page.html.**

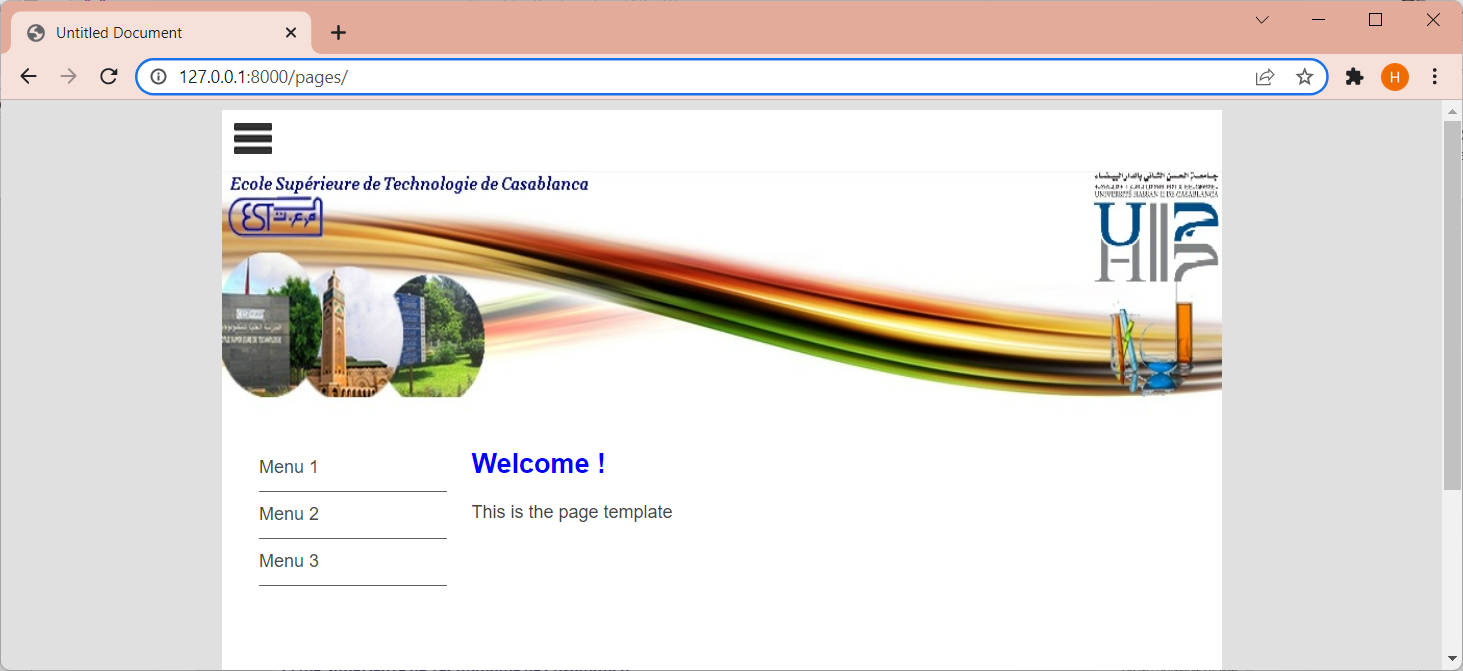
Maintenant que nous avons créé le template de page, nous devons modifier le fichier **views.py** pour afficher ce template (modifications en gras):

Une image contenant texte

Description générée automatiquement

Si vous exécutez le serveur de développement, et vous accédez à l’url : 

votre page d'accueil devrait ressembler à la figure ci-dessous. Notez qu'il est dit " This is the page template ", pas " This is the site template ". Cela démontre que, avec seulement quelques lignes de code, les templates de Django vous permettent de vous concentrer uniquement sur les choses qui sont différentes sur la page et d'ignorer les parties qui sont identiques.



**Synthèse des TP 03 et 04**

|  |
| --- |
| **Structure du projet** |
|  |

**==Fichier de configuration du projet : settings.py ====**

|  |
| --- |
| **D:/mydjango/mysite/mysite/settings.py** |
| |  | | --- | | **from** pathlib **import** Path | | **import** os | | **…........** | | INSTALLED\_APPS = [ | | **'pages.apps.PagesConfig'**, | | **'django.contrib.admin'**, | | **…........** | | ] | | **…........** | | ROOT\_URLCONF = **'mysite.urls'** | | **…........** | | TEMPLATES = [ | | { | | **'BACKEND'**: **'django.template.backends.django.DjangoTemplates'**, | | **'DIRS'**: [os.path.join(BASE\_DIR, **'mysite/templates'**)], | | **'APP\_DIRS'**: **True**, | | **…........** | | }, | | ] | | **…........** | | LANGUAGE\_CODE = **'fr-FR'** | |  | | **…........** | | STATIC\_URL = **'/static/'** | | STATICFILES\_DIRS = [os.path.join(BASE\_DIR, **'mysite/static'**)] | | **…........** | |

**==Fichiers du dossier interne du projet mysite ===**

|  |
| --- |
| **D:/mydjango/mysite/mysite/templates/base.html** |
| {% load static %} <!doctype **html**> <**html**> <**head**>  <**meta charset="utf-8"**>  <**title**>Untitled Document</**title**>  <**link href="{% static 'main.css' %}" rel="stylesheet" type="text/css"**> </**head**> <**body**>  <**div id="wrapper"**>   <**header id="header"**>  <**div id="logo"**><**img src="{% static 'logo.jpg' %}" alt=""**/></**div**>  <**div id="topbanner"**><**img src="{% static 'estc.jpg' %}"  width="800" height="200" alt=""**/>  </**div**>  </**header**>   <**aside id="leftsidebar"**>  <**nav id="nav"**>  <**ul**>  <**li**>Menu 1</**li**>  <**li**>Menu 2</**li**>  <**li**>Menu 3</**li**>  </**ul**>  </**nav**>  </**aside**>  <**section id="main"**>  {% block content %}  <**h1 style="color**:**red**;**"**>Welcome!</**h1**>  <**p**>This is the site template</**p**>  {% endblock content %}  </**section**>  <**footer id="footer"**>Copyright **&copy;** 2018 LPGLAASRI WebDesign</**footer**>  </**div**> </**body**> </**html**> |

|  |
| --- |
| **D:/mydjango/mysite/mysite/views.py** |
| Une image contenant texte  Description générée automatiquement |

|  |
| --- |
| **D:/mydjango/mysite/mysite/urls.py** |
|  |

**====Fichiers de l’applications pages ========**

|  |
| --- |
| **D:/mydjango/mysite/pages/templates/pages/page.html** |
| {% extends "base.html" %}  {% block content %} <**h1 style="color**: **blue**;**"**>Welcome !</**h1**> <**p**>This is the page template</**p**> {% endblock content %} |

|  |
| --- |
| **D:/mydjango/mysite/pages/apps.py** |
|  |

|  |
| --- |
| **D:/mydjango/mysite/pages/models.py** |
|  |

|  |
| --- |
| **D:/mydjango/mysite/pages/admin.py** |
|  |

|  |
| --- |
| **D:/mydjango/mysite/pages/views.py** |
|  |

|  |
| --- |
| **D:/mydjango/mysite/pages/urlss.py** |
|  |