# Développement web côté serveur

# Framework python-django

# Travaux pratiques n°4

# (Projet : essentiel\_templates)

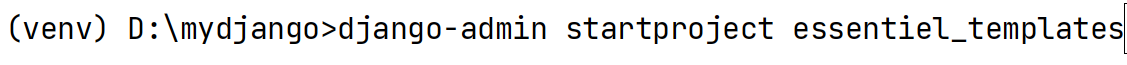
# L’essentiel sur le langage des templates

**Objectif :**

* Utiliser les fichiers statiques (css, javascript, images) dans un template
* Passer les données d’une view à un template
* Utiliser le langage des templates :
* Affichage des données dans un template
* Utiliser la structure alternative if…elif … endif
* Utiliser la structure de boucle for …endfor

|  |
| --- |
| 1. **Création de la Structure du projet ci-dessous** |
|  |

**Pour obtenir cette structure suivez les étapes ci-dessous :**

**1-Créer un nouveau projet django de nom essentiel\_templates avec la commande :  
**

**2-Créer le sous dossier static dans le dossier interne essentiel\_templates.**

**3-Créer un sous dossier css dans le dossier static. Le dossier css va contenir les fichiers de feuilles de styles .css.**

**4-Créer un sous dossier js dans le dossier static. Le dossier js va contenir les fichiers JavaScript .js.**

**5-Créer le sous dossier templates dans le dossier interne essentiel\_templates.**

**6-Modifier le fichier settings.py comme l’indique la figure ci-dessous.**

|  |
| --- |
| **Fichier settings.py** |
|  |

1. **Création de la Page d’accueil du projet**

|  |  |
| --- | --- |
| **1-Création du template index.html.**  Créer Le fichier **index.html** dans le dossier **templates** qui contiendra un ensemble de lien pour nous permettre d’accéder aux différentes pages du projet. Son format final est fourni ci-dessous. Mais comme nous n’avons pas encore créé les pages du projet, il faut replacer les balise "{% url ‘xxxxx’ %} " par "#" | |
| **E:\mydjango\essentiel\_templates\essentiel\_templates\templates\index.html** | |
| **Template : index.html (Version initiale)** | |
| <!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>Utilisation des templates</title> </head> <body> <h1>Essentiel sur le Langage des templates</h1> <h2><a href = "#">Utilisation des fichiers statiques css, js, images</a></h2> <h2><a href = "#">Affichage des variables</a></h2> <h2><a href = "#">Affichage des attributs d'un objet</a></h2> <h2><a href = "#">Utilisation des alternatives</a></h2> <h2><a href = "#">Utilisation des boucles</a></h2> </body> </html> | |
|  | |
| **Template : index.html (Version finale)** | |
|  | |
|  | |
| **2-Création de la vue**  Créer le fichier **views.py** dans le dossier interne du projet essentiel\_templates.  Puis créer la vue **index** qui permettra d'afficher le template **index.html** comme indiqué ci-dessous. | |
| **Fichier views.py** | |
| from django.shortcuts import render *#======================================================* def index(request):  return render(request, 'index.html') | |
|  | |
| **3-Ajout de l'url de la vue index**  Comme il s'agit de la page d'accueil du projet, l'url de la page est la chaine vide **''** comme le montre le fichier **urls.py** ci-dessous. | |
| **Fichier : urls.py** | |
| ] | |
|  | |
| **Lancer votre serveur en exécutant la commande :** | |
| **(venv) E:\mydjango\essentiel\_templates> python manage.py runserver** | |
| **Résultat d’exécution en ayant accédé à l’url : http://127.0.0.1:8000/** |
|  |

1. **Utilisation des fichiers statiques dans un template**

Pour utiliser un fichier static (images, css, JavaScript, …etc) dans un projet Django, il faut commencer par indiquer , dans le fichier **settings.py**, les dossiers qui contiennent ces fichiers statiques.

Nous devons créer le paramètre **STATICFILES\_DIRS** dont la valeur est une liste qui contient les dossiers où sont placés les fichiers statiques.

Pour garder le caractère dynamique des chemins, il est conseillé de concaténer les chemins relatifs de ces dossiers à la variable BASE\_DIR définie au tout début du fichier **settings.py.**

|  |
| --- |
| **Fichier settings.py** |
|  |

Pour charger un fichier statique dans un template, il faut utiliser la balise **static**. Nous devons commencer par charger la balise **static** au tout début du template HTML avec la balise   
**{% load static %}.**

L’exemple ci-dessous montre comment charger un fichier css et un fichier javascipt dans le template **temp\_static.html.**

|  |
| --- |
|  |

|  |
| --- |
| **Structure du projet** |
|  |
|  |
| **Fichier : valeur\_future.css** |
| **form** {  **font-family**: **Arial**, **Helvetica**, **sans-serif**;  **background-color**: **white**;  **width**: 600**px**;  **margin**: 0 **auto**;  **padding**: 1**em** 2**em** 1**em**;  **border**: 3**px solid blue**; }  **h1** {  **text-align**: **center**;  **color**: **blue**;  **margin-bottom**: 1**em**; }  **label** {  **float**: **left**;  **width**: 11**em**;  **text-align**: **right**; }  **input** {  **margin-left**: 1**em**;  **margin-bottom**: .5**em**;  **margin-right**: .5**em**;  **width**: 11**em**; }  **span** {  **color**: **red**; }  **table** {  **margin**: 0 **auto**;  **border-style**: **solid**;  **border-width**: 2**px**;  **border-color**: **blue**; } |
|  |
| **Fichier : tabstyle.css** |
| **table** {  **margin**: 0 **auto**;  **border**: **medium solid #6495ed**;  **border-collapse**: **collapse**;  **width**: 60%; }  **th** {  **font-family**: **monospace**;  **font-size**: 150%;  **border**: **thin solid #6495ed**;  **width**: 60%;  **padding**: 5**px**;  **background-color**: **#D0E3FA**; }  **td** {  **font-family**: **sans-serif**;  **border**: **thin solid #6495ed**;  **width**: 60%;  **padding**: 5**px**;  **text-align**: **center**;  **background-color**: **#ffffff**; }  **caption** {  **font-family**: **sans-serif**;  **font-size**: 200%;  **color**: **blue**; } |
|  |
| **Fichier : valeur\_future.js** |
| **var** *$* = **function**(id) {  **return *document***.getElementById(id); };  **var** *calculerVF* = **function**(investissement, taux, duree) {  **var** valeurFuture = investissement;  **for** (**var** i = 1; i <= duree; i++) {  valeurFuture += valeurFuture \* taux / 100;  }  valeurFuture = valeurFuture.toFixed(2);  **return** valeurFuture; };  **var** *traiterEntrees* = **function**() {  **var** investissement = *parseFloat*(*$*(**"investissement"**).**value**);  **var** taux = *parseFloat*(*$*(**"taux\_annuel"**).**value**);  **var** duree = *parseInt*(*$*(**"duree"**).**value**);  **var** toutesValides = **true**;   **if** (*isNaN*(investissement) || investissement <= 0 || investissement > 100000) {  *//$("investissement\_erreur").firstChild.nodeValue = "doit être > 0 et <= 100000";  $*(**"investissement\_erreur"**).**innerHTML** = **"doit être > 0 et <= 100000"**;  toutesValides = **false**;  } **else** {  *//$("investissement\_erreur").firstChild.nodeValue = "";  $*(**"investissement\_erreur"**).**innerHTML** = **""**;  }  **if** (*isNaN*(taux) || taux <= 0 || taux > 15) {  *//$("taux\_erreur").firstChild.nodeValue = "doit être > 0 et <= 15";  $*(**"taux\_erreur"**).**innerHTML** = **"doit être > 0 et <= 15"**;  toutesValides = **false**;  } **else** {  *//$("taux\_erreur").firstChild.nodeValue = "";  $*(**"taux\_erreur"**).**innerHTML** = **""**;  }  **if** (*isNaN*(duree) || duree <= 0 || duree > 50) {  *//$("duree\_erreur").firstChild.nodeValue = "Doit être > 0 et <= 50";  $*(**"duree\_erreur"**).**innerHTML** = **"Doit être > 0 et <= 50"**;  toutesValides = **false**;  } **else** {  *//$("duree\_erreur").firstChild.nodeValue = "";  $*(**"duree\_erreur"**).**innerHTML** = **""**;  }  **if** (toutesValides) {  *$*(**"valeur\_future"**).**value** = *calculerVF*(investissement, taux, duree);   *//document.write(tableauInvestissement(investissement,taux,duree));  $*(**"tableau"**).**innerHTML** = *tableauInvestissement*(investissement, taux, duree);  } };  **function** *tableauInvestissement*(investissement, taux, duree) {  **var** tab = **"<caption>Tableau d'investissement</caption>"** +  **"<tr><th>Année</th><th>Intérêts</th><th>ValeurFuture</th></tr>"**;  **var** valeurFuture = investissement;  **for** (**var** annee = 1; annee <= duree; annee++) {  **interet** = valeurFuture \* taux / 100;  valeurFuture += **interet**;  tab += **"<tr><td>"** + annee + **"</td><td>"** + **interet**.toFixed(2) +  **"</td><td>"** + valeurFuture.toFixed(2) + **"</td></tr>"**;   }  **return** tab; }   ***window***.**onload** = **function**() {  *$*(**"calculer"**).**onclick** = *traiterEntrees*;  *$*(**"investissement"**).focus(); }; |
|  |
| **Template : temp\_static.html** |
| {% load static %} <!DOCTYPE **html**> <**html**>  <**head**>  <**meta charset="UTF-8"**>  <**title**>Calcul d'épargne</**title**>  <**link rel="stylesheet" href="{% static 'css/valeur\_future.css' %}"**>  <**link rel="stylesheet" media="screen" type="text/css" title="Mon style" href=" {% static 'css/tabstyle.css' %}"** />   <**script src="{% static 'js/valeur\_future.js' %}"**></**script**> </**head**>  <**body**>   <**h1**>Calcul d'epargne</**h1**>  <**form**>  <**label for="investissement"**>Investissement initial:</**label**>  <**input type="text" id="investissement" value="10000"**>  <**span id="investissement\_erreur"**>**&nbsp;**</**span**><**br**>   <**label for="taux\_annuel"**>Taux d'interêt annuel:</**label**>  <**input type="text" id="taux\_annuel" value="5"**>  <**span id="taux\_erreur"**>**&nbsp;**</**span**><**br**>   <**label for="duree"**>Durée en années:</**label**>  <**input type="text" id="duree" value="10"**>  <**span id="duree\_erreur"**>**&nbsp;**</**span**><**br**>   <**label for="valeur\_future"**>Valeur Future:</**label**>  <**input type="text" id="valeur\_future" disabled**><**br**>   <**label**>**&nbsp;**</**label**>  <**input type="button" id="calculer" value="Calculer"**><**br**>  </**form**>  <**table id="tableau"**>    </**table**> </**body**>  </**html**> |
| **Fichier : views.py** |
|  |
|  |
| **Fichier : urls.py** |
|  |

**Remarque:** Par défaut Django trouve automatiquement tous les fichiers statiques inclus dans un dossier nommé static à l'intérieur du dossier d'une application.

|  |
| --- |
| * Application (Dossier de l’application)   + **static** ( le dossier doit être nommé exactement **static**)     - css       * fichier.css     - js       * fichier.js     - images       * fichier.png |

**Modifier le fichier index.html en changeant la ligne**

**<h2><a href = "#">Utilisation des fichiers statiques css, js, images</a></h2>**

Par

**<h2><a href = "{% url ‘view\_static’ %}">Utilisation des fichiers statiques css, js, images</a></h2>**

**Lancer votre serveur et tester**

1. **Insertion des données dans un template**

**Affichages des valeurs des variables**

Pour insérer des données dans un template, on utilise les doubles accolades {{ variable }}. Ces variables doivent au préalable être passées au paramètre **context** de la fonction **render**dans la vue :

|  |
| --- |
| views.py |
| **from django.shortcuts import render import datetime**  **def view\_variables(request):  vint = 10  vfloat = 10.5  vboolean = True  vstring = "Bonjour"  vdate = datetime.datetime.today()  dictionnaire = {  "vint" : vint,  "vfloat" : vfloat,  "vbool" : vboolean,  "vstring" : vstring,  "vdate" : vdate  }  return render(request, "temp\_variables.html", context = dictionnaire)** |
|  |
| urls.py |
|  |

On peut ensuite utiliser les variables passées au contexte à l'intérieur de notre fichier HTML :

|  |
| --- |
| Vector Directories Icon, Directories, Folders, Network Icon PNG and Vector  with Transparent Background for Free Download**template : temp\_variables.html** |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>Affichage des variables</title> </head> <body> <h1>Affichage des valeurs des variables</h1> <ul>  <li>vint : {{vint}}</li>  <li>vfloat : {{vfloat}}</li>  <li>vboolean : {{vboolean}}</li>  <li>vstring : {{vstring}}</li>  <li>vdate : {{vdate}}</li> </ul> </body> </html>** |
| Résultats Affichage |
| **Affichage des valeurs des variables**   * **vint : 10** * **vfloat : 10.5** * **vboolean :** * **vstring : Bonjour** * **vdate : Jan. 27, 2024, 6:32 p.m.** |

**Remarque :** Le nom d’une variable, contrairement à Python, ne peut pas commencer par le trait du soulignement. Ainsi le nom  \_vdate par exemple, provoquerait une erreur dans le fichier de template.

1. **Affichage des attributs d’un objet**

Il est possible d’afficher, à l'intérieur des doubles accolades, l'attribut d'un objet ou utiliser une méthode ou propriété d’un objet. Pour une méthode, il ne faut pas utiliser les parenthèses pour appeler la méthode. L’exemple suivant montre les étapes à suivre :

**Etape1**: créer le module utilitaires.py dans le dossier essentiel\_templatres interne, puis créer dedans la classe Etudiant ci-dessous

|  |
| --- |
| **E:/mydjango/essentiel\_templates/essentiel\_templates/Utilitaires.py** |
| **class Etudiant:  def \_\_int\_\_(self, matricule, nom, prenom, genre, note):  self.matricule = matricule  self.nom = nom  self.prenom = prenom  self.genre = genre  self.note = note   def nom\_complet(self):  return self.nom + " " + self.prenom   def genre\_entier(self):  if self.genre == 'M':  return 'Masculin'  else:  return 'Feminin'   def mention(self):  if self.note>=16:  return 'Tres Bien'  elif self.note>=14:  return 'Bien'  elif self.note>=12:  return 'Assez Bien'  elif self.note>=10:  return 'Passable'  else:  return 'Non Admis'** |
| **views.py** |
| **from django.shortcuts import render import datetime from .utilitaires import Etudiant**  **def view\_attributs(request):  etudiant = Etudiant("Etud001", "Bahroun", "Najib", "M", 12.5)  context = {"etud" : etudiant}  return render(request, "temp\_attributs\_objet.html", context)** |
|  |
| **Urls.py** |
| **from django.contrib import admin from django.urls import path from . import views  urlpatterns = [  path('admin/', admin.site.urls),  path('use\_static/', views.view\_static, name='view\_static'),  path('variables', views.view\_variables, name='view\_variables'),  path('attributs', views.view\_attributs, name='view\_attributs'),  path('', views.index, name='index')** |

On peut ensuite utiliser les variables passées au contexte à l'intérieur de notre fichier HTML :

|  |
| --- |
| Vector Directories Icon, Directories, Folders, Network Icon PNG and Vector  with Transparent Background for Free Download**template : temp\_attributs\_objet.html** |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>Affichage des attributs d'un objet</title> </head> <body> <h1>Affichage des attributs d'un objet</h1> <hr> <h2>Matricule : {{etud.matricule}}</h2> <h2>Nom et prenom : {{etud.nom\_complet}}</h2> <h2>Genre : {{etud.genre\_entier}}</h2> <h2>Note : {{etud.note}}</h2> <h2>Mention : {{etud.mention}}</h2> </body> </html>** |
|  |
| **http://127.0.0.1:8000/attributs** |
| **Affichage des attributs d'un objet**  **Matricule : Etud001**  **Nom et prenom : Bahroun Najib**  **Genre : Masculin**  **Note : 12.5**  **Mention : Assez Bien** |

1. **Les structures conditionnelles**

Pour créer une structure conditionnelle dans un template, on utilise la balise suivante :

|  |  |
| --- | --- |
| **If sans else** | **If avec else** |
| **{% if condition %}**  <h2>La condition est vraie !</h2>  **{% endif %}** | **{% if condition %}**  <h2>La condition est vraie !</h2>  **{% else %}**  <h2>La condition est fausse !</h2>  **{% endif %}** |
| {% if etudiant.note >= 10 %}  <h1>Admis !</h1>  {% endif %} | {% if etudiant.note >= 10 %}  <h1>Réussi</h1>  {% else %}  <h1>Non Admis</h1>  {% endif %} |

Notez l'utilisation d'une balise de fermeture **{% endif %}** qui permet d'indiquer la fin de la condition. Cette balise de fermeture est nécessaire car contrairement à Python, l'indentation dans un template HTML n'a aucune valeur. Là où Python utilise les niveaux d'indentations pour savoir les lignes de codes qui appartiennent à une structure conditionnelle, nous devons à l'intérieur d'un template, utiliser cette balise de fermeture.

Notez également l'absence des deux points à la fin de la condition.

On peut créer une structure conditionnelle plus avancées avec les balises elif et else :

|  |
| --- |
| {% if etudiant.note >= 16 %}  <h2>Mention = Très Bien </h2>  {% elif etudiant.note >= 14 %}  <h2>Mention = Bien </h2>  {% elif etudiant.note >= 12 %}  <h2>Mention = Assez Bien </h2>  {% elif etudiant.note >= 10 %}  <h2>Mention = Passable </h2>  {% else %}  <h2>Non Admis.</h2>  {% endif %} |

Tout le reste est similaire à Python. On peut ainsi utiliser les opérateurs **not**, **and** et **or** à l'intérieur des conditions.

**Exemple:**

|  |
| --- |
| **views.py** |
| **from django.shortcuts import render import datetime from .utilitaires import Etudiant**  ***#=====================================================================* def view\_alternatives(request):  etudiant = Etudiant("Etud002", "Chamsoun", "Meryme", "F", 19.5)  context = {"etud" : etudiant}  return render(request, "temp\_alternatives.html", context)** |
| **urls.py** |
| **from django.contrib import admin from django.urls import path from . import views  urlpatterns = [  path('admin/', admin.site.urls),  path('use\_static/', views.view\_static, name='view\_static'),  path('variables', views.view\_variables, name='view\_variables'),  path('attributs', views.view\_attributs, name='view\_attributs'),  path('alternatives', views.view\_alternatives,name='view\_alternatives'),  path('', views.index, name='index') ]** |
| **template : temp\_alternatives.html** |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>Alternatives</title> </head> <body> <hr> <h1>Langages des templates : balises if...elif...else</h1> <hr> <h2>Matricule : {{etud.matricule}}</h2> <h2>Nom & Prenom : {{etud.nom\_complet}}</h2> <hr> {% if etud.genre == 'M' %}  <h2>Genre : Masculin</h2> {% else %}  <h2>Genre : Feminin</h2> {% endif %} <hr> <h2>Note : {{etud.note}}</h2> {% if etud.note >= 16 %}  <h2>Mention : Tres Bien </h2> {% elif etud.note >= 14 %}  <h2>Mention : Bien</h2> {% elif etud.note >= 12 %}  <h2>Mention : Assez Bien</h2> {% elif etud.note >= 10 %}  <h2>Mention : Passable</h2> {% else %}  <h2>Mention : Non Admis</h2> {% endif %} <hr> </body> </html>** |
| **Résultats Exécution** |
|  |

1. **Les boucles**

Dans le langage des templates de Django, seule la boucle **for** est disponible (la boucle **while** n'existe pas).

Comme pour la balise **if**, la boucle **for** nécessite une balise de fermeture à cause de l'absence d'indentation dans les fichiers templates :

|  |
| --- |
| **{% for condition %}**  **<h1>corps de la boucle</h1>**  **{% endfor %}** |

La boucle **for** dispose également de plusieurs variables très pratiques que nous pouvons utiliser à l'intérieur des balises :

|  |  |
| --- | --- |
| **Variable** | **Signification** |
| **forloop.counter** : | L’itération actuelle de la boucle (index à partir de 1) |
| forloop.counter0 : | L’itération actuelle de la boucle (index à partir de 0) |
| forloop.revcounter : | Le nombre d’itérations à partir de la fin de la boucle (index à partir de 1) |
| forloop.revcounter0 : | Le nombre d’itérations à partir de la fin de la boucle (index à partir de 0) |
| forloop.first : | True si la boucle est dans sa première itération |
| forloop.last : | True si la boucle est dans sa dernière itération |
| forloop.parentloop : | Pour les boucles imbriquées, il s’agit de la boucle de niveau supérieur |

On peut également utiliser la boucle **for ... empty** pour afficher un élément dans le cas d'un itérable vide :

|  |
| --- |
| **{% for variable in iterable %}**  <h1>corps execute lorsque l’iterable n’est pas vide</h1>  **{% empty %}**  <h1>Exécuté lorsque l’itrable est vide</h1>  **{% endfor %}** |

**Exemples**

|  |
| --- |
| **urls.py** |
|  |
| **views.py** |
|  |
| **template : temp\_boucle.html** |
|  |
| **Résultats Exécution** |
|  |

**Les URL**

La balise url permet de référer à un chemin absolu d'un site. Cette balise utilise le paramètre **name** des chemins d'**URL** définis dans le fichier **urls.py** et est très utile pour utiliser des chemins dynamiques.

Considérant un fichier urls.py qui contient les chemins d'URL suivants :

|  |
| --- |
| # urls.py |
| from django.urls import path  urlpatterns = [  path('home/', views.index, name='index'),  path('blog/<str:slug>/', views.blog\_post, name='post')  ] |

Nous pouvons accéder au chemin absolu de ces URL dans un fichier template de la façon suivante :

**Pour un chemin d'URL sans paramètres d'URL (la vue "index")**

|  |
| --- |
| **<a href="{% url 'index' %}">Accueil</a>** |

Sera remplacé par :

|  |
| --- |
| <a href="home/">Accueil</a> |

**Pour un chemin d'URL avec paramètres (la vue "post")**

|  |
| --- |
| <a href="{% url 'post' slug='un-super-article' %}">Un super article</a> |

Sera remplacé par :

|  |
| --- |
| <a href="blog/un-super-article/">Un super article</a> |

L'intérêt d'utiliser la balise **url** est que nous pouvons à tout moment changer le chemin des URL (en remplaçant **'home/'** par **'index/'** par exemple) et tous nos chemins d'URL à l'intérieur des templates seront toujours valides (car on utilise le paramètre **name du chemin**, qui lui ne change pas).

**Héritage des templates**

Pour éviter de répéter du code HTML, on peut utiliser un système d'héritage de templates grâce aux balises extends et block.

On peut ainsi créer un fichier HTML que l'on viendra étendre avec un autre fichier HTML.

L'exemple le plus courant est de créer un fichier HTML de base qui contient toutes les balises HTML pour l'entête et le footer. On vient par la suite étendre ce fichier de base avec le contenu d'autres pages.

On crée donc un fichier **base.html** (le nom du fichier ainsi que le nom des blocs n'ont pas d'importance) :

|  |
| --- |
| Template de base : temp\_base.html |
| <!doctype html>  <html lang="fr">  <head>  {% block title %}<title>Page de base</title>{% endblock %}  </head>  <body>  {% block content %}  {% endblock %}  </body>  </html> |

On peut ensuite étendre ce fichier grâce à la balise **extends** et modifier le contenu des blocs avec la balise block :

|  |
| --- |
| Template : temp\_heritage**.html qui hérite de base.html** |
| {% extends 'base.html' %}  {% block title %}<title>Accueil du site</title>{% endblock %}  {% block content %}  <h1>Le contenu du site</h1>  {% endblock %} |

Le contenu que l'on ajoute dans les balises block viendra remplacer celui de la page qu'on **étends**. On remplace ainsi la balise <title>Page de base</title> du fichier base.html par <title>Accueil du site</title> dans le fichier index.html.

Dans un navigateur, la page **temp\_heritage.html** affiché contiendra ainsi le code HTML suivant :

|  |
| --- |
| <!doctype html>  <html lang="fr">  <head>  <title>Accueil du site</title>  </head>  <body>  <h1>Le contenu du site</h1>  </body>  </html> |