

TP6 – Premiers formulaires

Contexte et Objectifs

Pour le moment nous accédons à la base de données, et nous consultons les données, pour affichage. C'est de la « lecture seule ». L'idée est ici de :

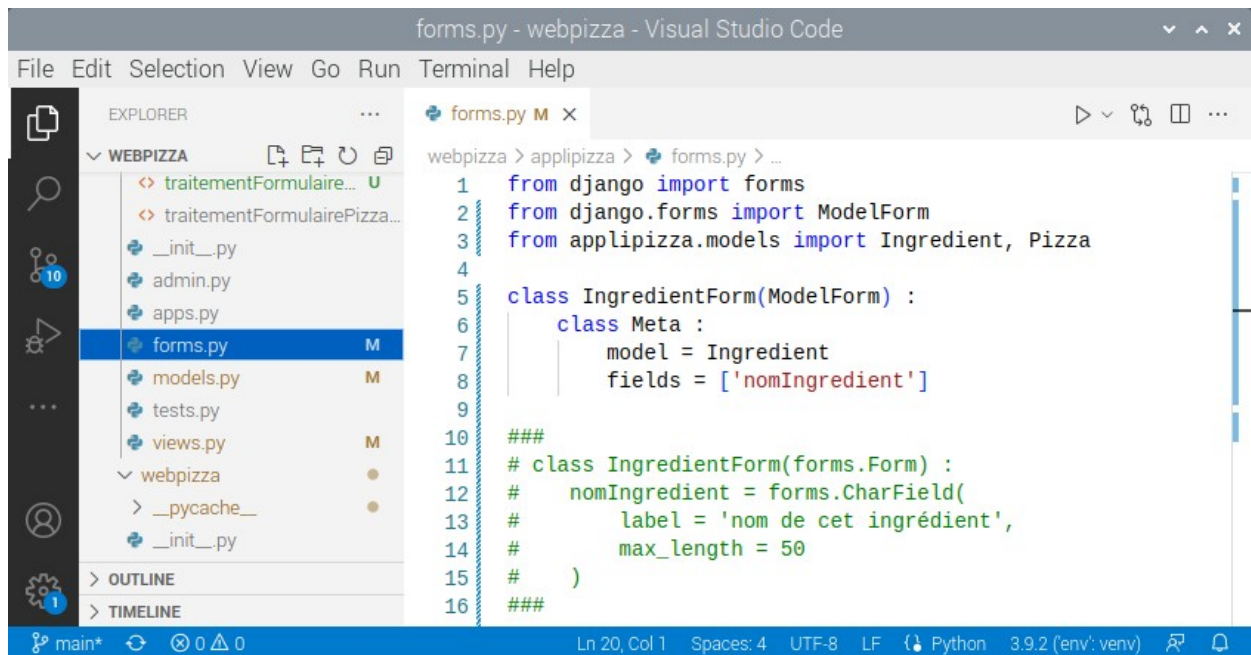
- construire un formulaire pour récupérer les données de futurs ingrédients,
- traiter l'envoi du formulaire et ainsi avoir une action d'écriture sur la base de données. Cette action se fera sans être connecté au site. Autrement dit, tout le monde est « administrateur ». Nous verrons la connexion plus tard.

Il y a une méthode complètement automatique pour créer un formulaire adapté aux attributs d'une classe. Nous allons l'utiliser pour les formulaires de création d'une pizza et d'un ingrédient. Nous verrons comment les faire « à la main » aussi.

Création du fichier forms.py

1. Au même niveau que views.py, créez un fichier forms.py. Nous allons y définir notre formulaire.
2. Dans ce fichier, importez la bibliothèque forms, puis de forms importez ModelForm, qui permet les formulaires automatiques. Créez la classe IngredientForm, qui hérite de ModelForm, et qui représentera le formulaire correspondant à un nouvel ingrédient.

Le code est vraiment des plus simples. La classe Meta fait le travail. En commentaires, la version non-automatique, où on déclare les champs équivalents à ceux de la classe.



```
forms.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
WEBPIZZA
  <> traitementFormulaire...
  <> traitementFormulairePizza...
  __init__.py
  admin.py
  apps.py
  forms.py M
  models.py M
  tests.py
  views.py M
  webpizza
    > __pycache__
    __init__.py
  > OUTLINE
  > TIMELINE

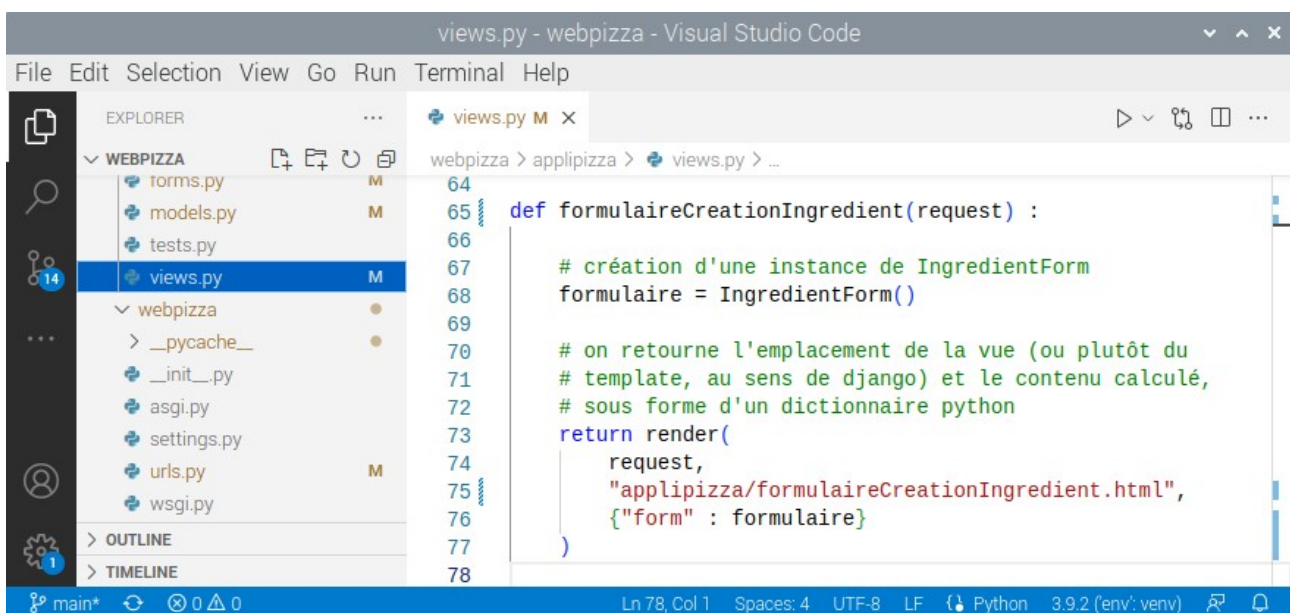
webpizza > applipizza > forms.py > ...
1 from django import forms
2 from django.forms import ModelForm
3 from applipizza.models import Ingredient, Pizza
4
5 class IngredientForm(ModelForm) :
6     class Meta :
7         model = Ingredient
8         fields = ['nomIngredient']
9
10 ###
11 # class IngredientForm(forms.Form) :
12 #     nomIngredient = forms.CharField(
13 #         label = 'nom de cet ingrédient',
14 #         max_length = 50
15 #     )
16 ###

Ln 20, Col 1 Spaces: 4 UTF-8 LF Python 3.9.2 (env: venv)
```

Création d'une « vue » dans le fichier views.py

1. Créez dans views.py la « vue » formulaireCreationIngredient :

- elle crée une instance de IngredientForm (pensez à insérer la classe)
- elle retourne un render qui indique un template (pas encore créé) **applipizza/formulaireCreationIngredient.html** en lui passant comme contenu le formulaire créé :



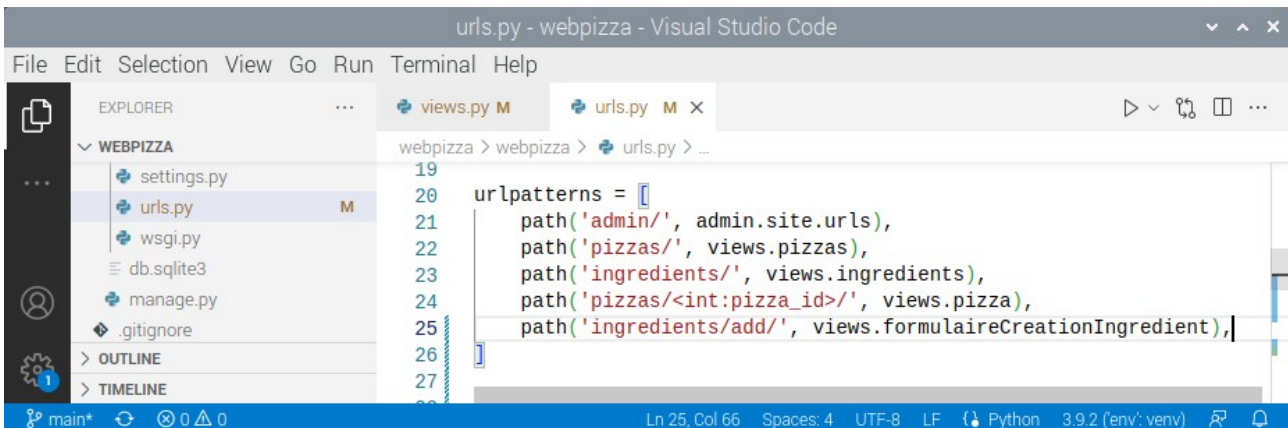
```
views.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
WEBPIZZA
  forms.py M
  models.py M
  tests.py
  views.py M
  webpizza
    > __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py M
    wsgi.py
  > OUTLINE
  > TIMELINE

webpizza > applipizza > views.py > ...
64
65 def formulaireCreationIngredient(request) :
66
67     # création d'une instance de IngredientForm
68     formulaire = IngredientForm()
69
70     # on retourne l'emplacement de la vue (ou plutôt du
71     # template, au sens de django) et le contenu calculé,
72     # sous forme d'un dictionnaire python
73     return render(
74         request,
75         "applipizza/formulaireCreationIngredient.html",
76         {"form" : formulaire}
77     )
78

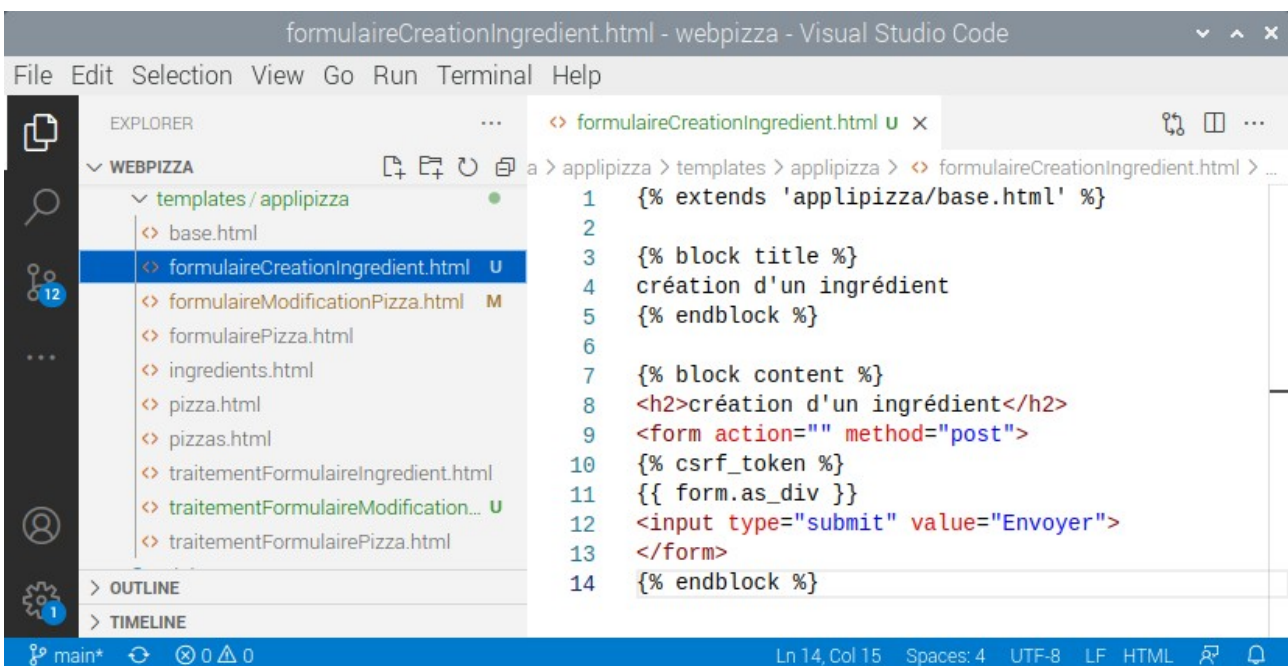
Ln 78, Col 1 Spaces: 4 UTF-8 LF Python 3.9.2 (env: venv)
```

2. Dans le fichier `urls.py`, définissez un path qui permet d'accéder à ce formulaire. Ce path pourra être de la forme `ingredients/add` et appellera la « vue » précédente.



```
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('pizzas/', views.pizzas),
23     path('ingredients/', views.ingredients),
24     path('pizzas/<int:pizza_id>', views.pizza),
25     path('ingredients/add/', views.formulaireCreationIngredient),
26 ]
27
```

3. Dans le répertoire `template/applipizza`, créez un fichier `formulaireCreationIngredient.html` qui aura la même base que les autres :



```
1 {% extends 'applipizza/base.html' %}
2
3 {% block title %}
4 création d'un ingrédient
5 {% endblock %}
6
7 {% block content %}
8 <h2>création d'un ingrédient</h2>
9 <form action="" method="post">
10 {% csrf_token %}
11 {{ form.as_div }}
12 <input type="submit" value="Envoyer">
13 </form>
14 {% endblock %}
```

On y retrouve l'intégration du formulaire `{{form}}` défini dans la « vue » `formulaireCreationIngredient`.

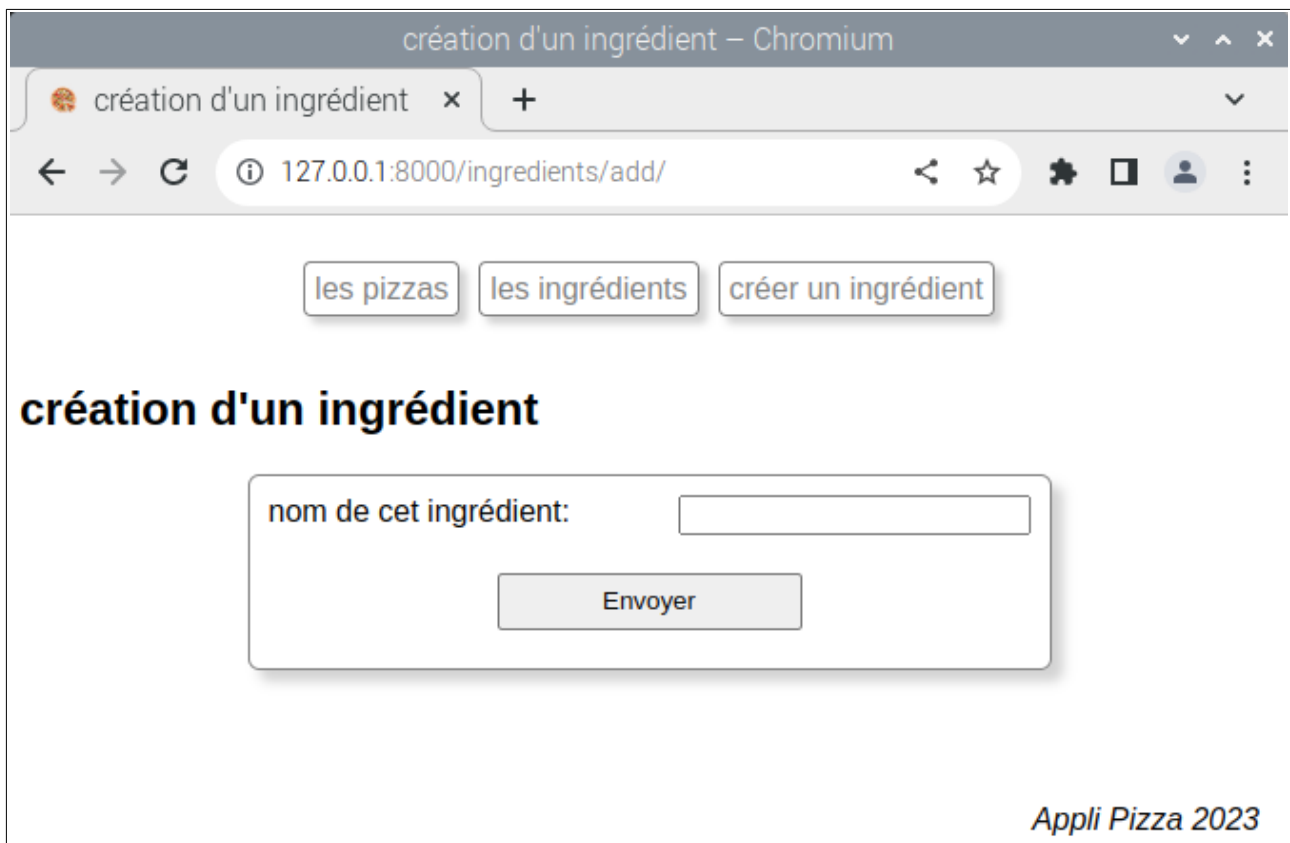
Le « `csrf_token` » est un « jeton » qui sert à lutter contre les failles CSRF. Nous pouvons laisser cela de côté, ce n'est pas notre thème.

<https://www.leblogduhacker.fr/faille-csrf-explications-contre-mesures/>

Pour le moment, on ne renseigne pas l'action du formulaire. Plus tard, l'action sera la création du nouvel ingrédient.

Si tout est bien fait, l'url 127.0.0.1:8000/ingredients/add/ doit amener au formulaire, qu'on peut remplir, mais dont l'envoi ne donnera rien, puisque le traitement du formulaire n'a pas encore été programmé.

Pensez à ajouter un lien dans le menu pour amener au formulaire :



The screenshot shows a web browser window titled 'création d'un ingrédient - Chromium'. The address bar displays '127.0.0.1:8000/ingredients/add/'. The page content includes a navigation menu with three buttons: 'les pizzas', 'les ingrédients', and 'créer un ingrédient'. Below the menu, the heading 'création d'un ingrédient' is displayed. A form box contains the label 'nom de cet ingrédient:' followed by a text input field and an 'Envoyer' button. The footer of the page reads 'Appli Pizza 2023'.

L'action lancée suite à l'envoi du formulaire

1. Créez dans views.py la « vue » `creerIngredient` qui va gérer le traitement des informations reçues via le formulaire :

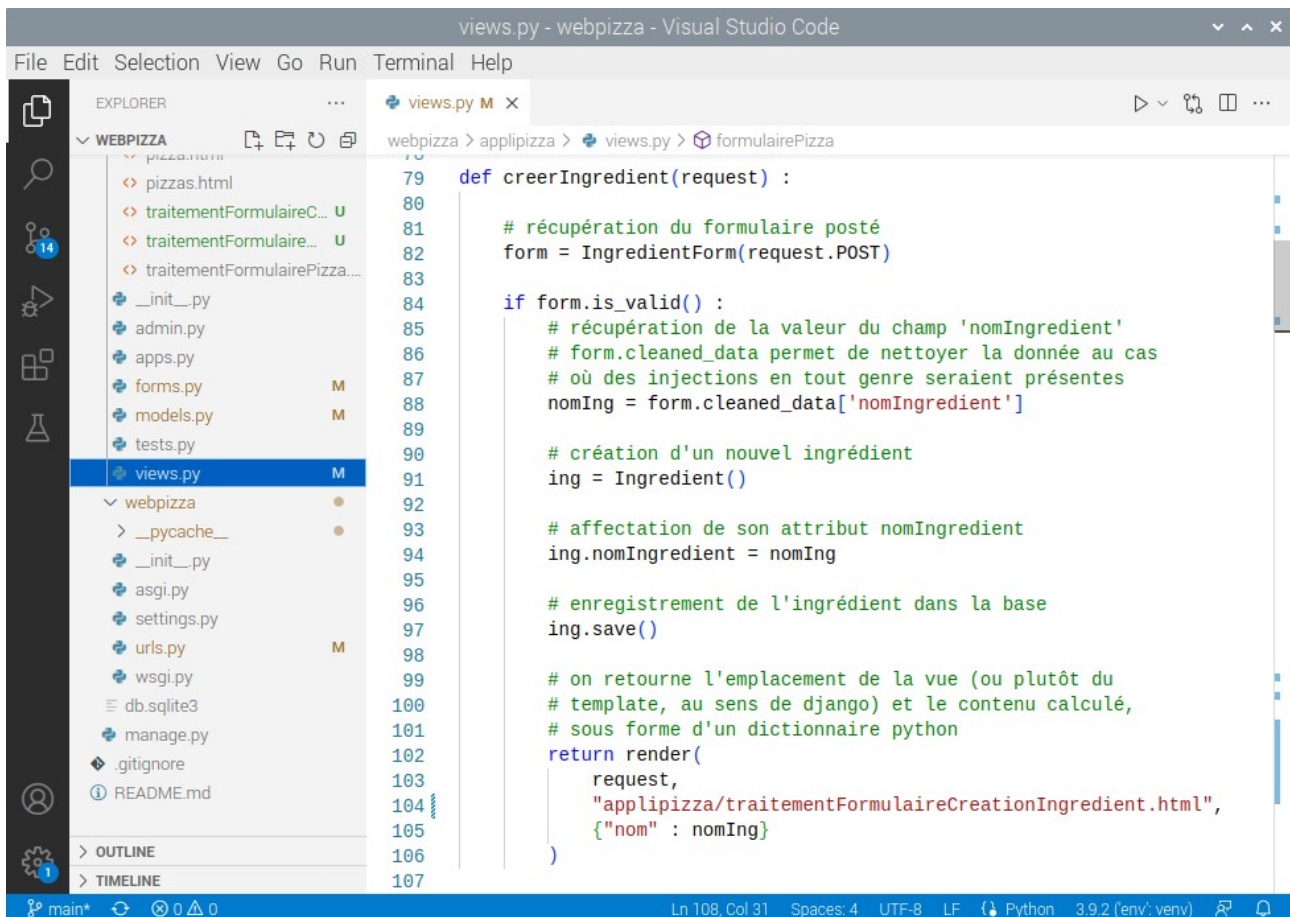
a. on récupère le formulaire :

```
form = IngredientForm(request.POST)
```

b. si le formulaire est « valide » (vérification automatique) alors on récupère le nom de l'ingrédient par :

```
nomIng = form.cleaned_data['nomIngredient']
```

- c. puis on crée une instance de la classe Ingredient, on lui donne comme nomIngredient la valeur récupérée, et on sauve cet Ingredient dans la base.
- d. Enfin, on retourne un render vers un template nommé applipizza/traitementFormulaireCreationIngredient.html, avec comme contenu le nom récupéré (template pas encore créé).

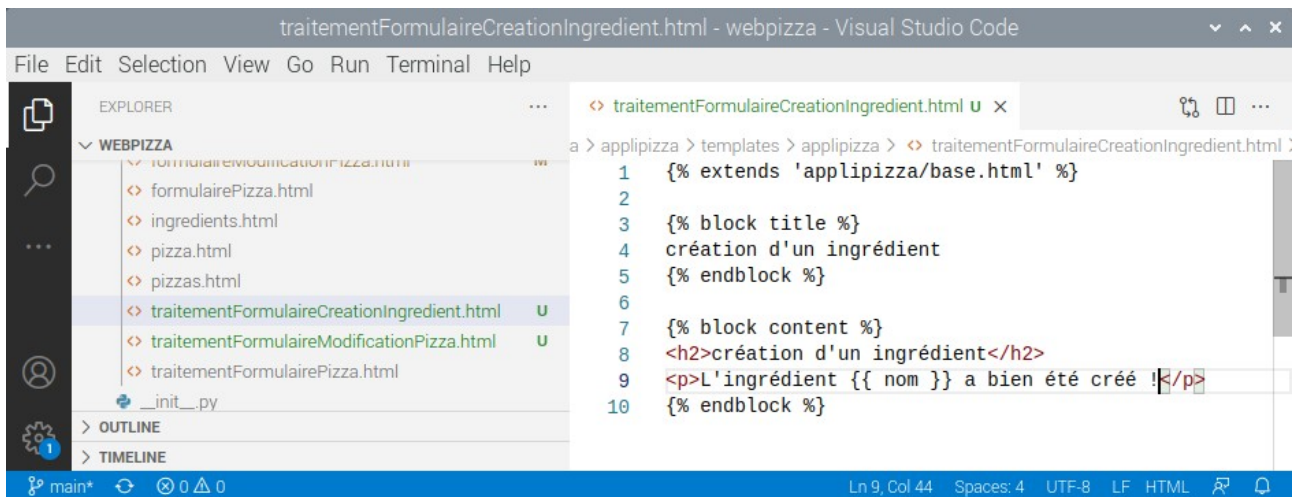


```
views.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
webpizza
  pizzas.html
  traitementFormulaireC...
  traitementFormulaire...
  traitementFormulairePizza...
  __init__.py
  admin.py
  apps.py
  forms.py
  models.py
  tests.py
  views.py
webpizza
  __pycache__
  __init__.py
  asgi.py
  settings.py
  urls.py
  wsgi.py
  db.sqlite3
  manage.py
  .gitignore
  README.md
OUTLINE
TIMELINE

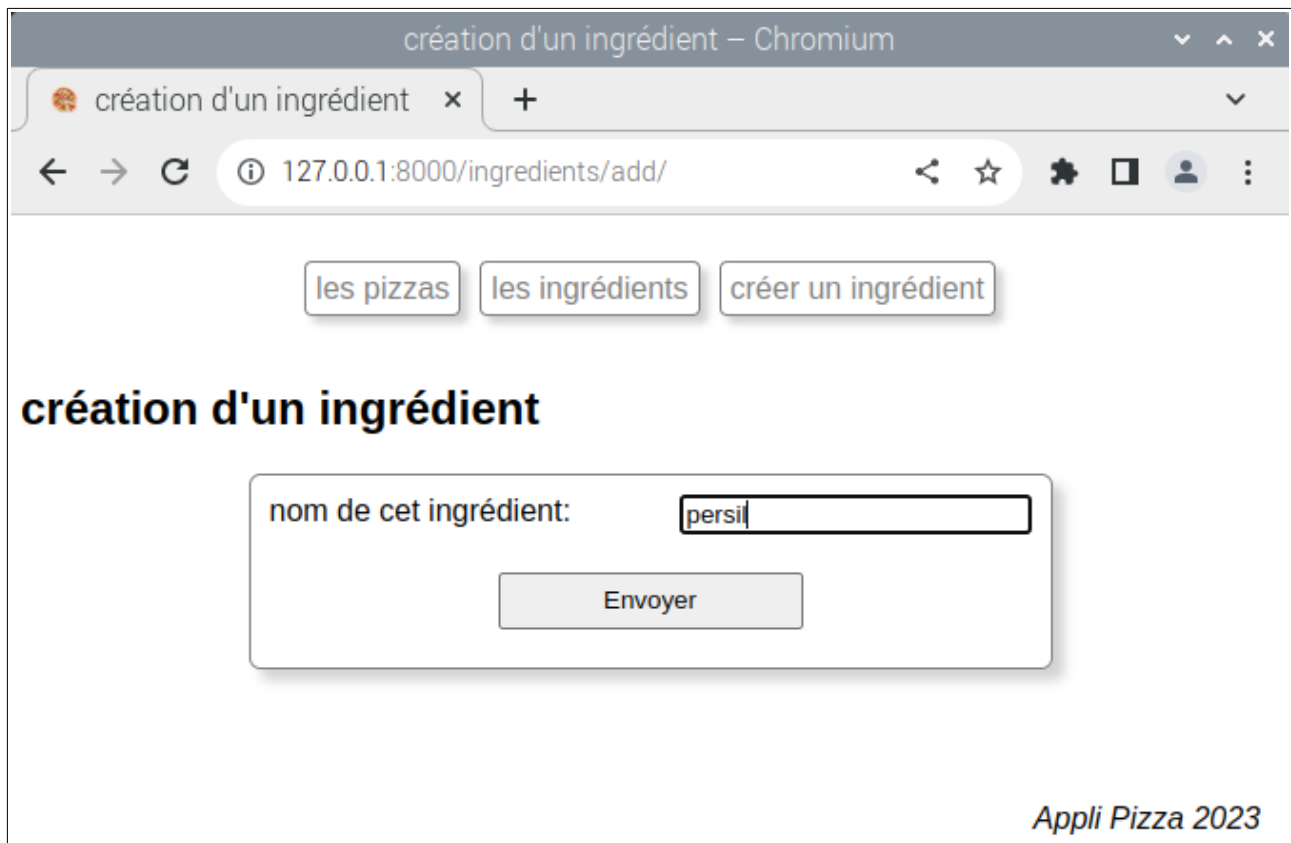
views.py
79 def creerIngredient(request) :
80
81     # récupération du formulaire posté
82     form = IngredientForm(request.POST)
83
84     if form.is_valid() :
85         # récupération de la valeur du champ 'nomIngredient'
86         # form.cleaned_data permet de nettoyer la donnée au cas
87         # où des injections en tout genre seraient présentes
88         nomIng = form.cleaned_data['nomIngredient']
89
90         # création d'un nouvel ingrédient
91         ing = Ingredient()
92
93         # affectation de son attribut nomIngredient
94         ing.nomIngredient = nomIng
95
96         # enregistrement de l'ingrédient dans la base
97         ing.save()
98
99         # on retourne l'emplacement de la vue (ou plutôt du
100         # template, au sens de django) et le contenu calculé,
101         # sous forme d'un dictionnaire python
102         return render(
103             request,
104             "applipizza/traitementFormulaireCreationIngredient.html",
105             {"nom" : nomIng}
106         )
107
```

2. Dans le fichier urls.py, définissez un path qui permet d'accéder à la page annonçant que le traitement a bien été effectué. Ce path sera de la forme ingredients/create/ et appellera la « vue » précédente.
3. Dans le template formulaireCreationIngredient.html, indiquez maintenant la bonne action : /ingredients/create/
4. Créez traitementFormulaireCreationIngredient.html, template qui hérite de base.html, comme tous les autres, et prévient juste que la création a été effectuée, et utilisant la variable de gabarit {{nom}} qui permet de rappeler le nom du nouvel ingrédient :



```
1 {% extends 'applipizza/base.html' %}
2
3 {% block title %}
4 création d'un ingrédient
5 {% endblock %}
6
7 {% block content %}
8 <h2>création d'un ingrédient</h2>
9 <p>L'ingrédient {{ nom }} a bien été créé !</p>
10 {% endblock %}
```

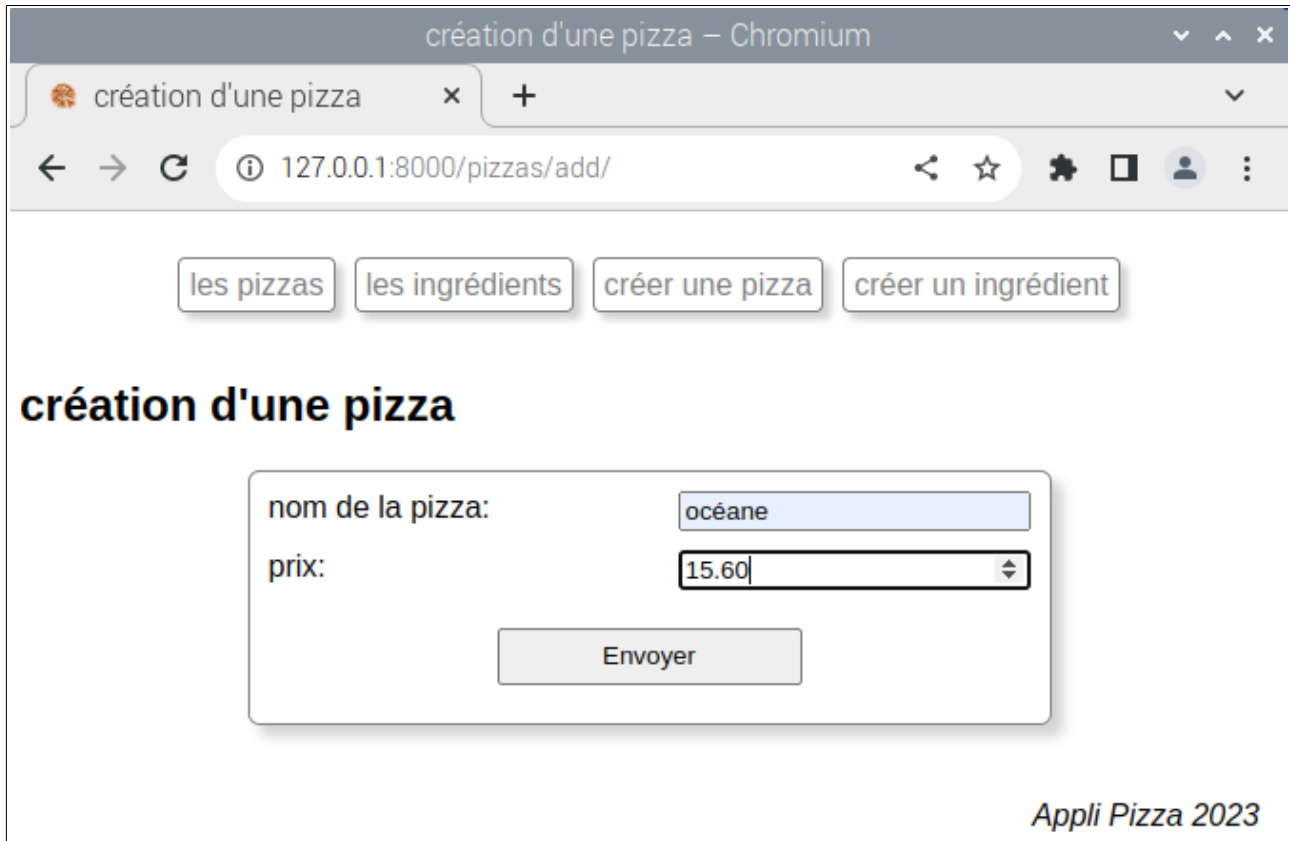
5. Testez que tout fonctionne. Vous pourrez aussi contrôler, dans `sqlitebrowser`, l'enrichissement progressif de la table des ingrédients.



Avez-vous bien tout assimilé ?

Recommencez tout, cette fois pour la création d'une nouvelle pizza (nom, et prix).

Voici un exemple de rendu :



The screenshot shows a web browser window titled "création d'une pizza – Chromium". The address bar displays "127.0.0.1:8000/pizzas/add/". The page features a navigation bar with four buttons: "les pizzas", "les ingrédients", "créer une pizza", and "créer un ingrédient". Below the navigation bar, the heading "création d'une pizza" is displayed. The main form contains two input fields: "nom de la pizza:" with the value "océane" and "prix:" with the value "15.60". A button labeled "Envoyer" is positioned below the form. The footer text "Appli Pizza 2023" is visible in the bottom right corner.

les pizzas les ingrédients créer une pizza créer un ingrédient

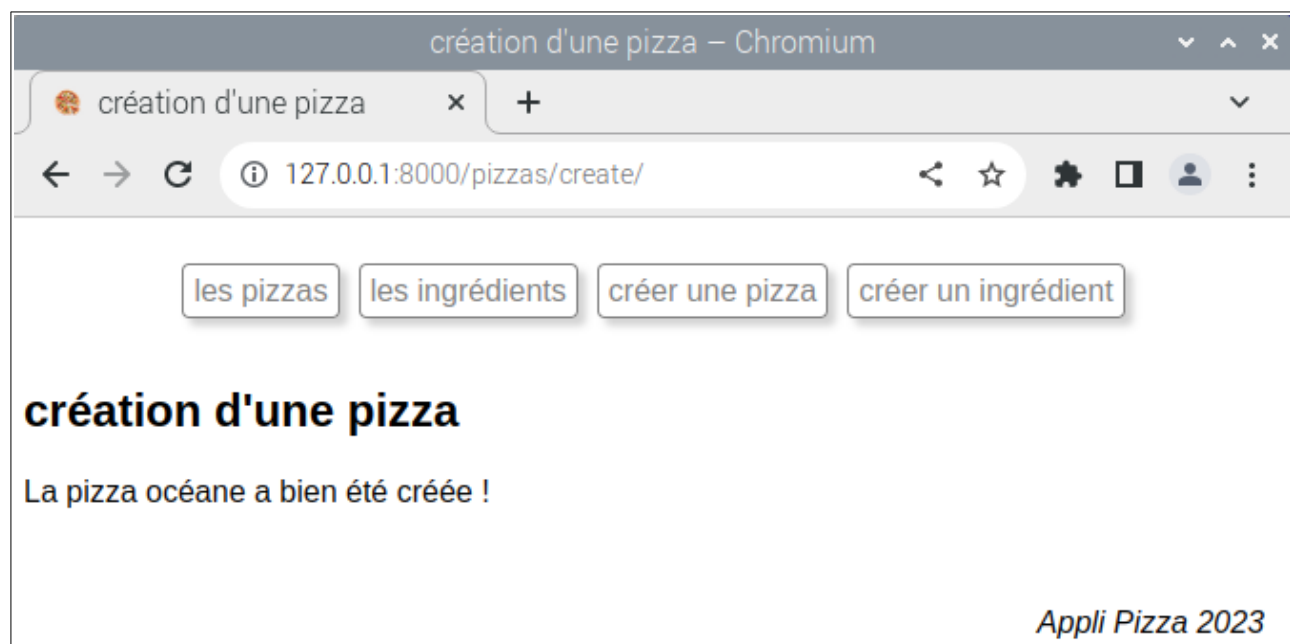
création d'une pizza

nom de la pizza: océane

prix: 15.60

Envoyer

Appli Pizza 2023



The screenshot shows the same web browser window, but the address bar now displays "127.0.0.1:8000/pizzas/create/". The navigation bar and heading remain the same. Below the heading, a message states "La pizza océane a bien été créée !". The footer text "Appli Pizza 2023" is still present in the bottom right corner.

les pizzas les ingrédients créer une pizza créer un ingrédient

création d'une pizza

La pizza océane a bien été créée !

Appli Pizza 2023



Remarque : on peut personnaliser l'affichage d'une pizza (voir capture précédente) de cette façon :

