LE FORMULAIRE DJANGO

Création d’un formulaire python

1. *Dans le dossier contenant les fichiers du site, créer un fichier de type python nommé* ***forms.py***

*Clic droit sur le dossier -> New -> Python file -> Name - > Python file*

**Dossier-Site/src/sous-dossier-site/forms.py**

1. *Dans ce fichier* **forms.py**
2. *on importe depuis Django le module* **forms** *(la classe, le type d’objet) qui permet de créer les formulaires*

**from** **django** **import** **forms**

1. *on créé un type de formulaire (un type d’objet, une classe) qui hérite module* **forms**

**class** **TypeFormulaire**(forms.Form):

HOMME = 'H'

FEMME = 'F'

SEXES = [

(HOMME, 'Homme'),

(FEMME, 'Femme')

]

*3. on créé les champs du formulaire (des variables et attributs) : des instances*

**variable = forms.TypeDeChamp(parametre=valeur, ou, attribut=valeur...)**

*NB. pour une variable équivalente à la balise ‘select’ (choices), une liste déroulante des choix possibles, on définit les options et leurs valeurs à l’intérieur de la fonction, dans un tuple*

**sexe = forms.CharField(choices=SEXES, default=HOMME)**

*NB. pour une variable équivalente à une case à cocher‘checkbox’, exemple :*

**case = forms.BooleanField(initial=True ou False)**

*Ce qu’on peut faire dans le formulaire à partir d’une console interactive python : le shell python*

1. *on peut créer le formulaire ou plutôt le stocker dans une variable (l’instancier):*

*>>> form = TypeDeFormulaire()*

1. *et le contenu (les variables, objets, instances) de ce formulaire :*

*>>> form = TypeDeFormulaire({‘cle’ : ‘valeur’})*

*>>> form = TypeDeFormulaire({‘label’ : ‘contenu’, label’ : ‘contenu’} )*

*>>> form = TypeDeFormulaire({‘email’ : ‘adresse@email.com’, ...})*

1. *on peut voir (afficher) ce formulaire sous la forme HTML*

*>>> print(form)*

1. *on peut vérifier si ce formulaire est valide (si les valeurs des attributs des champs indiquées dans la console correspondent aux valeurs prévues dans le code)*

*>>> form.is\_valid()*

*=> Entrée*

1. *pour réaliser tout cela, il faut :*

*Se positionner dans le dossier qui contient le fichier*  **manage.py,** *ouvrir un terminal et lancer shell (la console interactive) en tapant la commande :*

*python manage.py shell*

*Puis importer le formulaire (le type, la classe, le module)*

*>>> from website.forms import TypeDeFormulaire*

* *Entrée*

*Et on peut débuter les manipulations*

Après avoir créé le formulaire, il faut l’instancier (lui créer des variables) et l’afficher dans la vue (views.py)

1. *Dans le fichier* **views.py** *du site*

**from** **django.http** **import** HttpResponse

**from** **django.shortcuts** **import** **render**

*(1) on importe le formulaire créé (la classe, le type d’objet, le module)*

**from** **website.forms** **import** **TypeDeFormulaire**

*2) on définit le nom de la page (la fonction, méthode qui a pour rôle d’afficher ce formulaire)*

**def** nomDeLaPage(request):

1. *on stocke ce formulaire dans une variable (un attribut) : on instancie la classe*

**if request.method == ‘POST’**

**variable = forms.TypeDeChamp(parametre=valeur, ou, attribut=valeur...)**

*(3) on stocke ce formulaire dans une variable (un attribut) : on instancie la classe*

**form = TypeDeFormulaire**()

*(4) on renvoie le formulaire vers le fichier html (le templates) qui va l’afficher*

**return** render(request, dossier/fichier.html, **{'form': form}**)

1. *Dans le dossier* **templates** *du site,**créer un sous-dossier contenant le fichier html du formulaire*:

**src/templates/sous-dossier/nomDeLaPage.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Ma page de test</title>

</head>

<body>

*(1) choisir la méthode POST*

<**form** method="POST">

*(2) toujours laisser cette balise de protection contre les attaques*

{% **csrf\_token** %}

*(5) on affiche (les champs du) formulaire (entre les balises <p>)*

{{ form.as\_p }}

*(ou) entre les balises <ul> pour créer une liste*

{{ form.as\_ul }}

*(ou) entre les balises <table>, dans un tableau (le mieux aligné)*

<table>

{{ form.as\_table }}

</table>

*(6) rajoute le bouton d’envoie du formulaire*

<**input** type="submit" value="Submit">

</**form**>

</body>

</html>

Les composants de formulaire(widgets)

On peut personnaliser le formulaire (les champs) directement lorsqu’on créé la classe de formulaire (le Type de Formulaire) en spécifiant le type de widget (composant) que l’on souhaite utiliser pour chacun des champs.

(voir documentation)

**from** **django** **import** **forms**

**class** **TypeFormulaire**(forms.Form):

*exemple : un widget (composant) du type textarea :*

**texte = forms.CharField(widget=forms.Textarea())**

*exemple : un widget (composant) du type select avec des boutons radio :*

**choix = forms.ChoiceField(choices=CHOIX, widget=forms.RadioSelect())**

**ou**

**choix = forms.ChoiceField(choices=widget=forms.CheckboxSelectMultiple())**

**ou**

**choix = forms.ChoiceField(choices=CHOIX, widget=forms.SelectMultiple())**

*exemple : un widget (composant) pour masquer le mot de passe :*

**password = forms.CharField(widget=forms.PasswordInput())**

*Voir la documentation pour tous les widgets (composants)*

Récupérer les données du formulaire

1. ***Dans le fichier* forms.py**

*(1) spécifier la méthode (POST) d’envoi du formulaire*

<**form** **method**="POST">

*(2) toujours rajouter cette balise de protection contre les attaques*

{% **csrf\_token** %}

{{ form.as\_p }}

<**input** type="submit" value="Submit">

</**form**>

1. ***Dans la vue (la fonction) correspondante à ce formulaire* (fichier views.py)**

**def** nomDeLaPage(request):

1. *(Vérifier) si la méthode de requête HTTP utilisé dans le formulaire est du type POST*

**if request.method == ‘POST’ :**

*(2) conserver les données dans le formulaire tant qu’elles ne sont pas valides*

**variable = forms.TypeDeChamp(request.POST)**

*(5) si les données du formulaire son valides,*

**if variable.is\_valid() :**

*(6) Nettoyez ces données (affichez les avec cleaned\_data)*

**print(variable.cleaned\_data)**

*(7) puis afficher le réponse ci-dessous*

**Return HttResponse(‘Inscription réussie !’)**

*(3) sinon,*

**else :**

*(4) initialiser le formulaire : présenter un formulaire vierge*

**form = TypeDeFormulaire**()

*(4) on renvoie le formulaire vers le fichier html (le templates) qui va l’afficher*

**return** render(request, dossier/fichier.html, **{'form': form}**)

Validation personnalisée des champs

On va devoir rajouter des méthodes à l’intérieur de la classe de formulaire (Type de formulaire)

1. ***Dans la vue (la fonction) correspondante à ce formulaire* (forms.py)**

Définir une méthode clean\_nomChamp dans la classe du Formulaire

**def** clean\_champ(self):

1. accéder aux données du champs nettoyées dans la vue (views.py)
2. vérifier s’il y a une erreur dans le champ.
3. S’il y a une erreur, afficher un message d’erreur

Lier le formulaire à un modèle (une table de base de données)

1. Dans **models.py** créer un model (une table, une classe) en lien avec le formulaire
2. Dans **forms.py** créer le type de formulaire (la classe), mais cette fois-ci, le faire hériter par **(forms.ModelForm)**
3. Dans **views.py** créer le type de formulaire (la classe), mais cette fois-ci, le faire hériter par **(forms.ModelForm)**
4. importer (la classe de) formulaire créé
5. l’instancier comme précédemment
6. et retourner le formulaire dans un contexte comme précédemment