

django

Le framework Django

Les formulaires

Formation Haïti
08/05/27 - 11/05/12

Formulaire d'ajout d'une couche

- Dans le fichier forms.py

```
from django import forms
from layers.models import Layer

class AddLayerForm(forms.ModelForm):
    class Meta:
        model = Layer
        fields = '__all__'
```

Formulaire d'ajout d'une couche

- Dans le fichier views.py avec une nouvelle vue

```
form = AddLayerForm(request.POST or None)
```

Formulaire d'ajout d'une couche

- Dans un fichier template

```
<form action="{% url "upload" %}" method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Submit" />
</form>
```

Remarques

- CSRF : Cross-Site Request Forgeries
 - Champ caché avec une valeur unique
 - Vérifie que l'utilisateur à bien afficher la page du formulaire et évite une redirection pirate
- Rendu d'un formulaire, plusieurs templates possibles :
 - as_p, as_table, as_ul

Notre propre règle de validation

Fonction qui se nomme clean_nom_du_champ

```
def clean_author(self):
    author = self.cleaned_data['author']
    if "root" in author:
        raise forms.ValidationError(
            "Vous ne devez pas utiliser votre compte root !")

    return author
```

Formulaire indépendant d'un modèle

- Notre propre formulaire de contact

```
class ContactForm(forms.Form):  
    sujet = forms.CharField(max_length=100)  
    message = forms.CharField(widget=forms.Textarea)  
    envoyeur = forms.EmailField(label="Votre adresse mail")  
    renvoi = forms.BooleanField(help_text="Cochez si vous souhaitez  
obtenir une copie du mail envoyé.", required=False)
```

Formulaire indépendant d'un modèle

- Récupération des champs dans la vue

```
if form.is_valid():
    sujet = form.cleaned_data['sujet']
    message = form.cleaned_data['message']
    envoyeur = form.cleaned_data['envoyeur']
    renvoi = form.cleaned_data['renvoi']

    envoi = True
```

Les signaux

Les signaux par défaut

Signal	Description	Arguments
pre_save	Envoyé avant qu'une instance de modèle ne soit enregistrée.	sender : le modèle concerné instance : l'instance concerné using : alias de la BDD raw: bool
post_save	Envoyé après qu'une instance de modèle a été enregistrée.	sender , instance , created , bool si l'enregistrement a été correct
pre_delete	Envoyé avant qu'une instance de modèle ne soit supprimée.	sender , instance , using

Les signaux

- Nous avons des fichiers liés à un enregistrement :
- Une couche dans la BDD -> shapes files, tiff ...

```
from django.db.models.signals import post_delete  
  
post_delete.connect(fonction_suppression_des_fichiers, sender=Layer)  
  
  
def ma_fonction_de_suppression(sender, instance, **kwargs):  
    print 'Je dois supprimer les fichiers annexes à la couche.'
```

Vos propres signaux

- Nous pouvez aussi créer vos propres signaux

```
from django.dispatch import Signal
```

```
# Nouveau signal  
# On fait des traitements sur nos données. On souhaite être notifié  
# quand le traitement est terminé.  
shapefile_created = Signal(providing_args=[ "path" , ] )
```

```
# Connexion de mon signal  
shapefile_created.connect(display_layer)
```

```
# Dans mon fonction de d'analyse spatiale  
shapefile_created.send(sender=self, path=path_to_my_new_shapefile)
```