

TP N°7 (TP N°6 suite)

Python-Django FrameWork

Formulaires : Concepts Avancés

Objectifs du TP :

- Création d'un formulaire avancé

Le formulaire de contact que nous avons construit dans le dernier TP est une utilisation commune mais très simple d'un formulaire de site Web. Une autre utilisation courante des formulaires consiste à collecter des informations auprès de l'utilisateur et à les enregistrer dans la base de données.

L'utilisation de formulaires pour collecter et enregistrer des données auprès des utilisateurs est si courante que Django a une classe de formulaire spéciale qui permet de créer des formulaires à partir de modèles Django beaucoup plus facilement – model forms.

Avec les formulaires de modèle, vous créez un modèle Django, puis créez un formulaire qui hérite de la classe *ModelForm* de Django. Pour montrer comment tout cela fonctionne, nous allons créer un nouveau **modèle** et un nouveau **formulaire de modèle**.

LPGLAASRI, étant une «société» de conception de sites Web, a besoin d'un moyen pour permettre aux clients de soumettre une demande de devis. Une fois que la demande de devis a été soumise, elle doit être sauvegardée dans la base de données pour que le personnel de la société puisse examiner les besoins du client et revenir au client avec un devis.

Rappelez-vous que les meilleures pratiques de Django stipulent que chaque application Django ne doit faire qu'une chose. Pour obtenir le résultat souhaité, nous devons commencer par créer une nouvelle application Django.

Le processus est le même que celui que nous avons suivi pour créer l'application pages:

1. Créer l'application **quotes**;
2. Créer le model **Quote** et l'ajouter à la base;
3. Ajouter le model **Quote** à l'admin de Django. Cette fois-ci, nous allons également peaufiner l'interface utilisateur d'administration pour faciliter la gestion des données de devis.
4. Créez le formulaire de modèle *QuoteForm* pour collecter les informations de demande de devis auprès de l'utilisateur
5. Ajouter une nouvelle vue pour gérer le formulaire;
6. Créer le template du formulaire; et
7. Ajoutez notre nouvelle view et notre nouveau formulaire à notre *urls.py* et mettez à jour le template de site pour le lier au formulaire de devis.

La tâche peut paraître difficile, mais le processus est simple et s'appuie sur ce que nous avons déjà appris.

Création de l'application Quotes

Comme mentionné dans l'introduction, une application Django ne devrait faire qu'une chose à la fois. La collecte et la gestion des demandes de devis émanant des utilisateurs de notre

site sont une application très différente de l'affichage des pages du site. Nous devons donc créer une nouvelle application Django pour les devis.

Assurez-vous que votre environnement virtuel est en cours d'exécution, puis exécutez la commande suivante depuis le répertoire externe \mysite2019

```
(venv2019) C:\Users\hassouni\Django2019\mysite2019>python manage.py startapp quotes
```

Si vous avez exécuté cette commande correctement, vous aurez maintenant une nouvelle application (*quotes*) au même niveau que l'application *pages* dans votre répertoire de projet.

Pendant que vous y êtes, ajoutez un dossier \ *uploads* au même niveau que votre application *quotes*. Vous aurez besoin de ce dossier pour stocker les fichiers de devis téléchargés.

```
(venv2019) C:\Users\hassouni\Django2019\mysite2019>md uploads
```

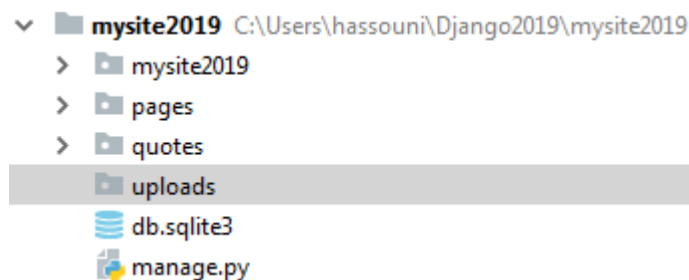
```
(venv2019) C:\Users\hassouni\Django2019\mysite2019>dir
```

Volume in drive C has no label.

Volume Serial Number is A853-3C6A

Directory of C:\Users\hassouni\Django2019\mysite2019

```
06/03/2019 13:41 <DIR>      .
06/03/2019 13:41 <DIR>      ..
06/03/2019 13:43 <DIR>      .idea
17/02/2019 16:41          139 264 db.sqlite3
07/02/2019 21:40          557 manage.py
20/02/2019 22:44 <DIR>      mysite2019
28/02/2019 11:41 <DIR>      pages
06/03/2019 13:39 <DIR>      quotes
06/03/2019 13:41 <DIR>      uploads
                2 File(s)      139 821 bytes
                7 Dir(s) 195 124 158 464 bytes free
```



```
▼ mysite2019 C:\Users\hassouni\Django2019\mysite2019
  > mysite2019
  > pages
  > quotes
  uploads
  db.sqlite3
  manage.py
```

Pour enregistrer l'application avec votre projet Django, nous devons ajouter la configuration de l'application *quotes* à votre liste *INSTALLED_APPS* dans notre fichier de paramètres *settings.py* (modifications en gras):

C:\Users\hassouni\Django2019\mysite2019\mysite2019\settings.py

```
INSTALLED_APPS = [  
    'pages.apps.PagesConfig',  
    'quotes.apps.QuotesConfig',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

Creation du modèle Quote

Lorsque Django a créé l'application *quotes* pour vous, il a également créé un nouveau fichier *models.py* pour l'application. Ouvrez ce nouveau fichier et entrez ce qui suit:

```

1 from django.db import models
2 from django.contrib.auth.models import User
3
4 STATUS_CHOICES = (
5     ('NEW', 'New Site'),
6     ('EX', 'Existing Site'),
7 )
8
9 PRIORITY_CHOICES = (
10    ('U', 'Urgent - 1 week or less'),
11    ('N', 'Normal - 2 to 4 weeks'),
12    ('L', 'Low - Still Researching'),
13 )
14
15 class Quote(models.Model):
16     name = models.CharField(max_length=100)
17     position = models.CharField(max_length=60, blank=True)
18     company = models.CharField(max_length=60, blank=True)
19     address = models.CharField(max_length=200, blank=True)
20     phone = models.CharField(max_length=30, blank=True)
21     email = models.EmailField()
22     web = models.URLField(blank=True)
23     description = models.TextField()
24     sitestatus = models.CharField(max_length=20, choices=STATUS_CHOICES)
25     priority = models.CharField(max_length=40, choices=PRIORITY_CHOICES)
26     jobfile = models.FileField(upload_to='uploads/', blank=True)
27     submitted = models.DateField(auto_now_add=True)
28     quotedate = models.DateField(blank=True, null=True)
29     quoteprice = models.DecimalField(decimal_places=2, max_digits=7, blank=True, default=0)
30     username = models.ForeignKey(User, blank=True, null=True, on_delete=models.CASCADE)
31
32 def __str__(self):
33     return str(self.id)

```

C'est un modèle beaucoup plus grand que notre modèle de page, mais les principes fondamentaux sont les mêmes. Passons en revue certaines des parties les plus importantes:

Ligne 2. Nous allons accéder à la base de données *User* pour lier un utilisateur à une instance du modèle *quote*, nous devons donc importer la classe *User* à partir de *django.contrib.auth.models*. Nous fournirons davantage d'informations sur ce sujet ultérieurement.

Lignes 4 à 7. STATUS_CHOICES est un tuple de tuples à deux éléments (deux tuples) qui se traduira par une liste déroulante d'options sur le formulaire modèle dans l'admin et sur le site web. Cela pourrait également être entré comme une liste de tuples, cependant, la meilleure pratique consiste à utiliser des tuples car ils sont immuables (ne peuvent pas être changés). Le premier élément du tuple est la valeur qui sera enregistrée dans la base de données, le deuxième élément est le nom lisible par l'homme.

Lignes 9 to 13. PRIORITY_CHOICES est le même que STATUS_CHOICES; il sera traduit par une liste d'options dans les formulaires.

Ligne 24. Le champ sitestatus a un attribut supplémentaire *choices*. Lorsque vous définissez l'attribut choice, Django remplacera le widget TextInput standard par le widget Select. Le

widget Sélectionner affiche une liste déroulante contenant le contenu de l'attribut choices, c'est-à-dire **STATUS_CHOICES**.

Ligne 25. Le champ **priority** définit également l'attribut **choice**. Dans ce cas, la liste déroulante du widget Select est fournie par **PRIORITY_CHOICES**.

Ligne 26. Nous utilisons un **FileField** pour la première fois dans un modèle. Nous fournissons l'attribut **upload_to**, donc Django sait où placer les fichiers téléchargés.

Ligne 27. Nous définissons l'attribut **auto_now_add** sur **True**. Cela enregistrera automatiquement la date et l'heure actuelles dans le champ **submitted**.

Ligne 28. Django ne définit jamais un champ de date à une valeur vide. Par conséquent, pour autoriser le champ **quotedate** à être vide, nous devons également définir l'attribut **null** sur **True** pour autoriser Django à enregistrer une entrée Null lorsque **quotedate** est vide.

Ligne 30. Nous avons ajouté un lien de clé étrangère (Foreign Key) au modèle **User**. Les valeurs vides ne sont pas autorisées pour les clés étrangères, nous avons donc défini l'attribut **null** sur **True**. L'attribut **on_delete** est requis. Si vous ne le définissez pas, Django émettra une erreur. Nous définissons l'attribut **on_delete** sur **CASCADE**, ce qui causera la suppression de toutes les entrées associées dans les autres tables.

Maintenant que nous avons créé le modèle, vérifions que tout a été correctement saisi avec la commande de gestion **check**:

```
(venv2019) C:\Users\hassouni\Django2019\mysite2019>python manage.py check
```

Si le modèle a été entré correctement, vous devriez voir quelque chose comme ceci dans sur écran:

```
System check identified no issues (0 silenced).
```

Ensuite, nous devons créer et exécuter les migrations avec successivement la commande **makeimgrations** et **migrate**:

```
(venv2019) C:\Users\.....\mysite2019>python manage.py makemigrations
```

```
Migrations for 'quotes':
```

```
quotes\migrations\0001_initial.py
```

```
- Create model Quote
```

```
(venv2019) C:\Users\hassouni\....\mysite2019>python manage.py migrate
```

```
Operations to perform:
```

```
Apply all migrations: admin, auth, contenttypes, pages, quotes, sessions
```

```
Running migrations:
```

```
Applying quotes.0001_initial... OK
```

Et c'est tout pour le modèle Quote. Maintenant, nous devons l'ajouter à l'admin afin que nous puissions gérer les enregistrements du modèle.

Ajout du modèle Quote à Django Admin

Pour pouvoir gérer les enregistrements du modèle, nous devons ajouter le modèle à l'admin. C'est assez simple; Commençons par enregistrer une classe d'administration simple:

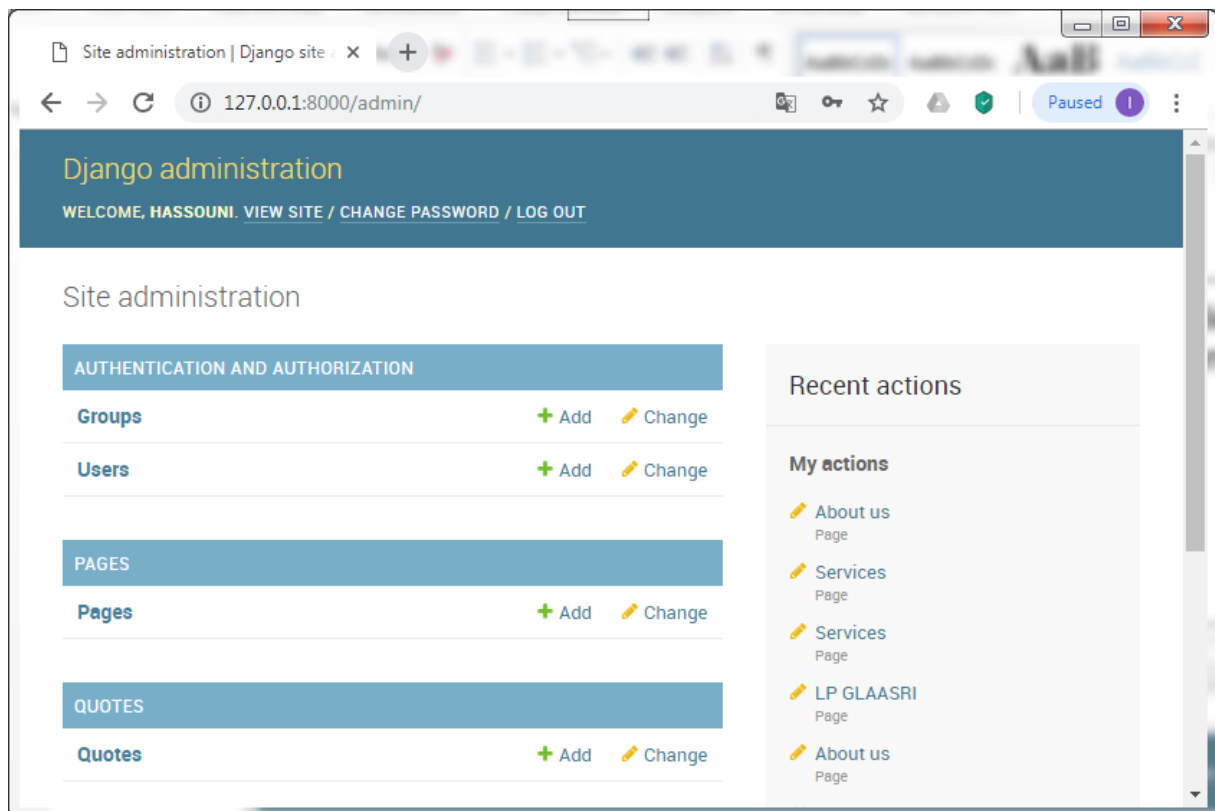
```
C:\Users\hassouni\Django2019\mysite2019\quotes\admin.py
```

```
from django.contrib import admin  
from .models import Quote
```

```
admin.site.register(Quote)
```

Nous ne reviendrons pas sur ça parce que ça devrait être familier maintenant. Le but de ce code est de s'assurer que l'admin fonctionne correctement avec votre modèle. Lancez le serveur de développement et naviguez jusqu'à

<http://127.0.0.1:8000/admin/>



et essayez d'ajouter un nouveau enregistrement.

Si tout a fonctionné comme planifié, vous devriez voir un formulaire vierge avec tous vos champs prêts à remplir. Le formulaire est présenté ci-dessous.

The screenshot shows the Django administration interface for adding a quote. The browser address bar indicates the URL is `127.0.0.1:8000/admin/quotes/quote/add/`. The page title is "Django administration" and the user is logged in as "HASSOUNI". The breadcrumb trail shows the path: Home > Quotes > Quotes > Add quote.

The form is titled "Add quote" and contains the following fields:

- Name:
- Position:
- Company:
- Address:
- Phone:
- Email:
- Web:
- Description:
- Site status:
- Priority:
- Job file: (Choose File) No file chosen
- Quoted date: Today
- Quote price: 0
- Username:

At the bottom of the form, there are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

Le formulaire est très grand et ne tient pas sur une seule page. Comment pouvons-nous rendre ce formulaire plus gérable? Mettons à profit ce que nous avons appris au TP3 (Modèles et Admin Django) et améliorons l'interface de gestion du modèle. (changements en gras):

`C:\Users\hassouni\Django2019\mysite2019\quotes\admin.py`

```

1 from django.contrib import admin
2 from .models import Quote
3
4 class QuoteAdmin(admin.ModelAdmin):
5     list_display = ('id', 'name', 'company', 'submitted', 'quotedate', 'quoteprice')
6     list_filter = ('submitted', 'quotedate')
7     readonly_fields = ('submitted',)
8     fieldsets = (
9         (None, {
10             'fields': ('name', 'email', 'description')
11         }),
12         ('Contact Information', {
13             'classes': ('collapse',),
14             'fields': ('position', 'company', 'address', 'phone', 'web')
15         }),
16         ('Job Information', {
17             'classes': ('collapse',),
18             'fields': ('sitestatus', 'priority', 'jobfile', 'submitted')
19         }),
20         ('Quote Admin', {
21             'classes': ('collapse',),
22             'fields': ('quotedate', 'quoteprice', 'username')
23         }),
24     )
25
26 admin.site.register(Quote, QuoteAdmin)

```

Actualisez l'admin après ces modifications et Ajouter un enregistrement. Le formulaire est beaucoup plus clair et devrait ressembler à la figure ci-dessous.

The screenshot shows the Django administration interface for adding a new quote. The browser address bar indicates the URL is `127.0.0.1:8000/admin/quotes/quote/add/`. The page title is "Django administration" and the user is logged in as "HASSOUNI". The breadcrumb trail is "Home > Quotes > Quotes > Add quote".

The "Add quote" form consists of the following fields:

- Name:** A text input field.
- Email:** A text input field.
- Description:** A large text area for a detailed description.

Below the main form fields are three collapsible sections, each with a "Show" button:

- Contact Information (Show):** Contains fields for position, company, address, phone, and web.
- Job Information (Show):** Contains fields for sitestatus, priority, jobfile, and submitted.
- Quote Admin (Show):** Contains fields for quotedate, quoteprice, and username.

At the bottom of the form are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

Il y a quelques éléments significatifs dans la classe QuoteAdmin, donc nous allons travailler avec eux avec quelques exemples et quelques captures d'écran pour illustrer ce qui se passe.

Entrez quelques enregistrements de test: il vous suffit de remplir les champs où le nom du champ est en gras. N'oubliez pas d'étendre **Contact Information** et **Job Information** pour entrer des données supplémentaires. Ne vous inquiétez pas du panneau **Quote Admin** à ce stade.

The screenshot shows a web browser window with the URL `127.0.0.1:8000/admin/quotes/quote/add/`. The page title is "Add quote | Django site admin". The form contains the following fields:

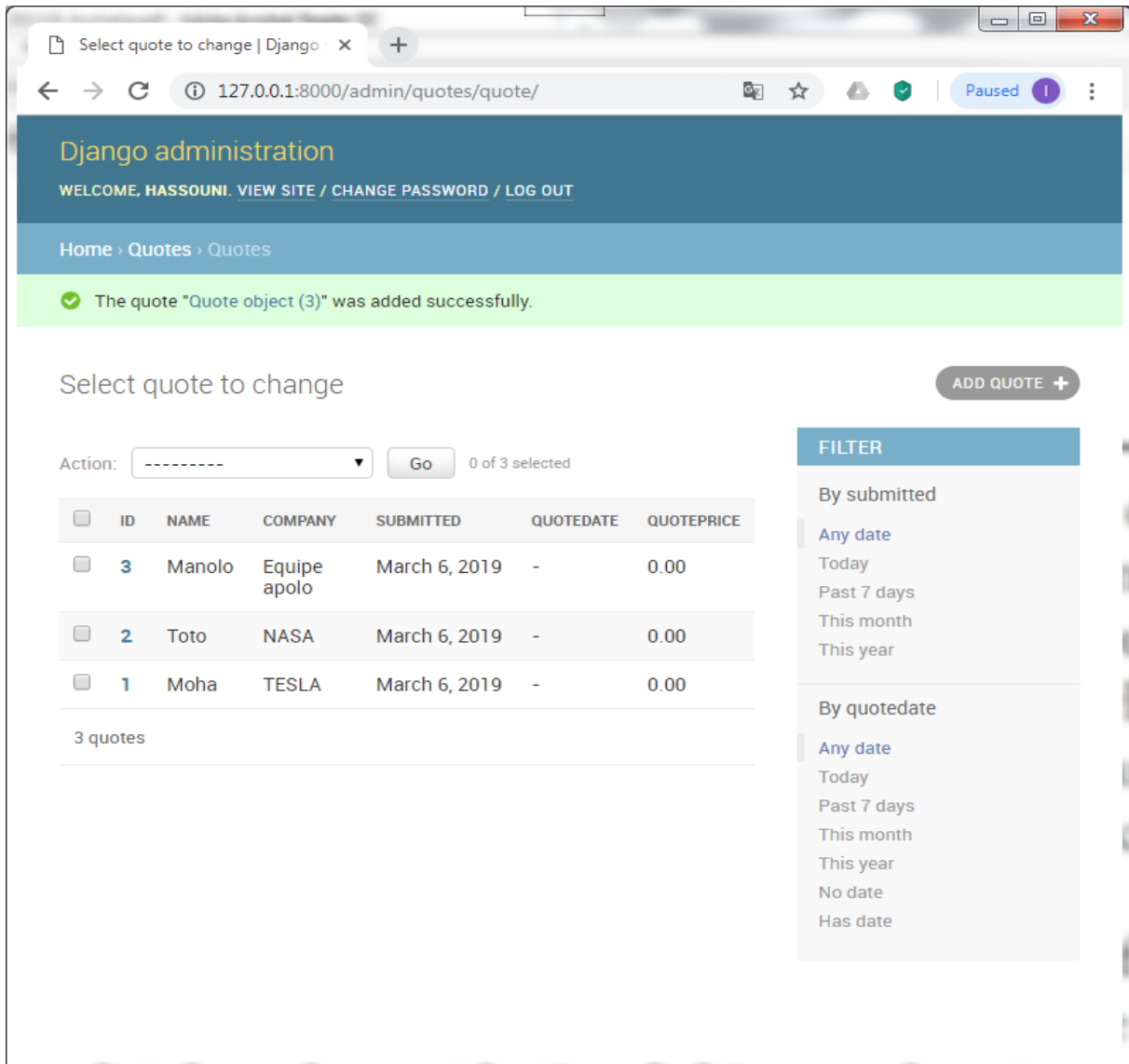
- Add quote**
 - Name:**
 - Email:**
 - Description:**
- Contact Information (Hide)**
 - Position:**
 - Company:**
 - Address:**
 - Phone:**
 - Web:**
- Job Information (Hide)**
 - Sitestatus:**
 - Priority:**
 - Jobfile:** No file chosen
 - Submitted:** -
- Quote Admin (Show)**

Une fois que vous avez ajouté quelques enregistrements, les listes devraient ressembler à la figure ci-dessous. A partir de cette capture d'écran, nous pouvons voir ce que la classe **QuoteAdmin** a ajouté à la mise en forme et à la gestion des modèles:

list_display : détermine quelles colonnes sont affichées dans la liste

list_filter : indique à l'admin Django les champs à utiliser pour réaliser des filtres (les filtres sont à droite de l'écran). Le filtrage devient utile lorsque le nombre d'enregistrements dans

vosre base de données augmente. Par exemple, vous pouvez filtrer la liste pour afficher uniquement les quotes qui ont été soumis ce mois-ci.



Le regroupement des champs réductibles dans le formulaire d'admin est configuré avec l'option `fieldsets` dans notre classe `QuoteAdmin`. L'option `fieldsets` contrôle la disposition des pages `add` et `edit` dans l'admin. C'est un ensemble de deux-tuples(`<name>`, `<field_options>`). L'ordre des deux tuples gouverne l'ordre dans lequel les groupes de champs sont affichés dans chaque section de la page de l'admin.

`name` est le nom donné au groupe. Si `name` est `None`, alors aucun titre de groupe ne sera affiché. `field_options` est un dictionnaire d'options pour le groupe de champs dans chaque section du formulaire d'affichage.

Voyons voir comment cela fonctionne dans la pratique. Voici notre premier fieldset:

```
(None, {
    'fields': ('name', 'email', 'description')
}),
```

Ce fieldset affichera les champs `name`, `email` et `description` comme les trois premiers champs sur le formulaire d'admin, sans le titre de section (voir figure ci-dessous).

Add quote

Name:

Email:

Description:

None entraîne l'affichage des champs du groupe sans titre de groupe.

Les trois groupes de champs suivants regroupent les champs restants en trois sections:

1. Contact Information;
2. Job Information; et
3. Quote Admin.

Chacun de ces fieldsets a une option classes supplémentaires qui applique la classe `collapse` au fieldset. `collapse` est une classe intégrée spéciale qui utilise JavaScript pour appliquer un accordéon à un ensemble de champs. Vous pouvez voir l'effet de cette classe dans la figure ci-dessous où chacun des trois derniers champs apparaissant est replié lorsque le formulaire est affiché pour la première fois. Pour développer la section, il vous suffit de cliquer sur le lien ([Show](#)).

Contact Information ([Show](#))

Job Information ([Show](#))

Quote Admin ([Show](#))

Les options fieldset regroupent les champs de formulaire et la classe "collapse" les place dans un panneau JavaScript repliable et pratique

Une dernière chose avant que nous n'allions de l'avant: ouvrez le groupe **Job Information** et vous remarquerez que le champ **Submitted** n'est pas modifiable (Voir Figure ci-dessous).

Job Information (Hide)

Sitestatus:

Priority:

Jobfile: No file chosen

Submitted: March 6, 2019

Comme il s'agit d'un champ horodaté (créé en définissant `auto_now_add=True` sur le modèle), il ne peut pas être modifié dans l'admin. Si nous essayions de l'afficher dans le formulaire, nous aurions une erreur. Nous avons donc défini l'option `readonly_fields` sur le modèle admin

(`QuoteAdmin`) pour inclure le champ **Submitted** afin qu'il puisse être affiché dans le formulaire en lecture seule.

Maintenant que notre application **quotes** est opérationnelle et que l'administrateur du modèle est trié, Il est temps de créer le formulaire type, ainsi que la view et le Template associés, ce qui nous permettra d'afficher le formulaire sur le site Web et de recueillir les demandes de devis des clients.

Le processus est le suivant:

- 1- Créez le formulaire modèle (model form) **QuoteForm** pour collecter les informations de demande de devis auprès de l'utilisateur.
- 2- Ajouter une nouvelle view pour gérer le formulaire;
- 3- Créer le Template du formulaire; et
- 4- Ajoutez notre nouvelle view et le nouveau formulaire à notre fichier `urls.py` et mettez à jour le Template du site afin de créer un lien vers le formulaire Quote.

Création du formulaire de l'application Quote

C'est là que la puissance de la classe `ModelForm` de Django brille vraiment - créer un formulaire pour le modèle est une tâche presque triviale. Créez un nouveau fichier `forms.py` dans le dossier de votre application `quotes` et entrez le code suivant:

```
1 from django import forms
2 from django.forms import ModelForm
3 from .models import Quote
4
5 class QuoteForm(ModelForm):
6     required_css_class = 'required'
7     class Meta:
8         model = Quote
9         fields = [
10             'name', 'position', 'company', 'address',
11             'phone', 'email', 'web', 'description',
12             'sitestatus', 'priority', 'jobfile'
13         ]
```

Ces quelques lignes de code suffisent à Django pour créer un formulaire pour votre modèle qui affichera tout le code HTML nécessaire sur la page, validera vos champs de formulaire et transmettra les informations à votre view. Il y a des choses à noter, alors regardons-les maintenant:

- **Ligne 2.** Nous importons la classe `ModelForm`, qui fera tout le travail lourd pour nous.
- **Ligne 5.** Notre classe `QuoteForm` hérite de la classe `ModelForm`.
- **Ligne 6.** C'est une option pratique de la classe `ModelForm` qui ajoute une classe CSS à nos champs obligatoires. Nous allons utiliser cette classe pour ajouter un astérisque (*) aux champs obligatoires dans le template de formulaire.

- **Ligne 7.** La classe `ModelForm` a une classe interne `Meta` que nous utilisons pour transmettre dans les métadonnées ce dont la classe `ModelForm` a besoin pour rendre notre formulaire:

- ✓ **Ligne 8.** Le modèle sur lequel notre formulaire est basé
- ✓ **Ligne 9.** Les champs du modèle à afficher sur le formulaire.

Ajout d'une View à l'application Quote

La *view* de quotes s'appuie également sur ce que nous avons appris dans les TP précédents. Nous allons appeler la nouvelle view `quote_req`, alors allons-y et ajoutons le code de la view au fichier `views.py` dans notre application quotes:

```
C:\Users\hassouni\Django2019\mysite2019\quotes\views.py
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3
4 from .models import Quote
5 from .forms import QuoteForm
6 from pages.models import Page
7
8 def quote_req(request):
9     submitted = False
10    if request.method == 'POST':
11        form = QuoteForm(request.POST, request.FILES)
12        if form.is_valid():
13            form.save()
14            return HttpResponseRedirect('/quote/?submitted=True')
15    else:
16        form = QuoteForm()
17        if 'submitted' in request.GET:
18            submitted = True
19
20    return render(request, 'quotes/quote.html',
        {'form': form,
         'page_list': Page.objects.all(),
         'submitted': submitted})
```

Cette vue est fonctionnellement identique à celle de notre formulaire de contact, à l'exception du fait que nous avons supprimé le code pour l'envoi par courrier électronique des données du formulaire et que nous l'avons remplacé par la méthode **`form.save()`** afin de sauvegarder les données du formulaire dans notre base de données (ligne 13).

Il y a un changement important à noter: à la ligne 11, nous avons ajouté **`request.FILES`** aux arguments passés à la classe `QuoteForm`. C'est ainsi que nous pouvons récupérer les informations de téléchargement de fichiers à partir de la réponse.

Creation du template du formulaire de l'application Quote

Il est maintenant temps de créer le Template pour notre formulaire. Nous hériterons du modèle de base du site. Le formulaire sera donc très similaire au modèle de formulaire de contact. Il existe une différence majeure: le formulaire HTML doit être un formulaire en

plusieurs parties (multipart form : enctype="multipart/form-data") pour que nous puissions télécharger un fichier avec le formulaire.

Commencez par créer un nouveau dossier \ *templates* dans le dossier de l'application *quotes*, ajoutez un autre dossier à l'intérieur de celui-ci (appelé \ *quotes*) et ajoutez le code suivant:

C:\Users\hassouni\Django2019\mysite2019\quotes\templates\quotes

```
1 {% extends "base.html" %}
2
3 {% block title %}Quote Request{% endblock title %}
4
5 {% block sidenav %}
6     {% for page in page_list %}
7         <li><a href="{{ page.permalink }}">{{ page.title }}</a></li>
8     {% endfor %}
9 {% endblock sidenav %}
10
11 {% block content %}
12 <h1>Quote Request</h1>
13
14 {% if submitted %}
15     <p class="success">
16         Your quote was submitted successfully. Thank you.
17     </p>
18
19 {% else %}
20     <form action="" enctype="multipart/form-data" method="post" novalidate>
21     <table>
22         {{ form.as_table }}
23     <tr>
24         <td>&nbsp;</td>
25         <td><input type="submit" value="Submit"></td>
26     </tr>
27 </table>
```

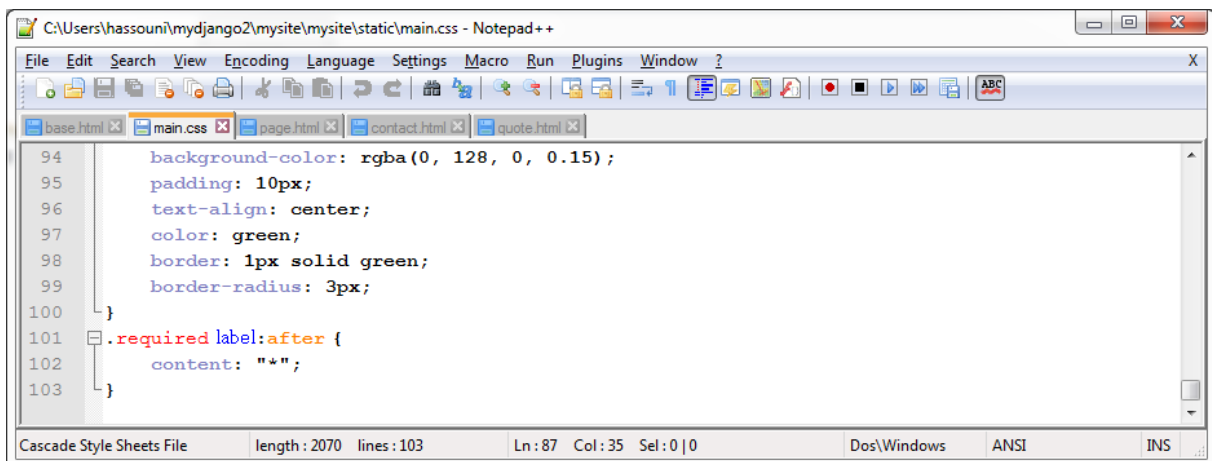
```
28 {% csrf_token %}
29 </form>
30 {% endif %}
31 {% endblock content %}
```

Ce code devrait être simple, et ne doit pas nécessiter d'aller dans les détails. Le seul changement par rapport à ce que nous avons vu précédemment est la ligne 20 où nous définissons l'attribut `enctype="multipart/form-data"` pour gérer le téléchargement du fichier.

Pendant que nous travaillons sur le modèle de formulaire, nous devons ajouter un petit ajustement au fichier `main.css` afin que nos étiquettes de champs obligatoires aient un astérisque ajouté à l'étiquette:

```
# ajouter à la fin du fichier \static\main.css

.required label:after {
content: "*";
}
```



Lier le formulaire QuoteForm

Maintenant que nous avons terminé notre modèle, l'admin et le formulaire, il ne nous reste que configurer correctement les fichiers `urls.py` du projet et de l'application `quotes`, et ajouter un lien au formulaire de `quotes` dans le fichier `base.html`.

Modifier le fichier `urls.py` du projet pour inclure le fichier `urls.py` de l'application `quotes` comme ci-dessous (**ligne en gras**) :

```
C:\Users\hassouni\Django2019\mysite2019\mysite2019\urls.py
```

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('quote/', include('quotes.urls')),
    path('', include('pages.urls')),
]
```

Créer le fichier urls.py de l'application quotes avec le contenu ci-dessous.

\quotes\urls.py (nouveau fichier)

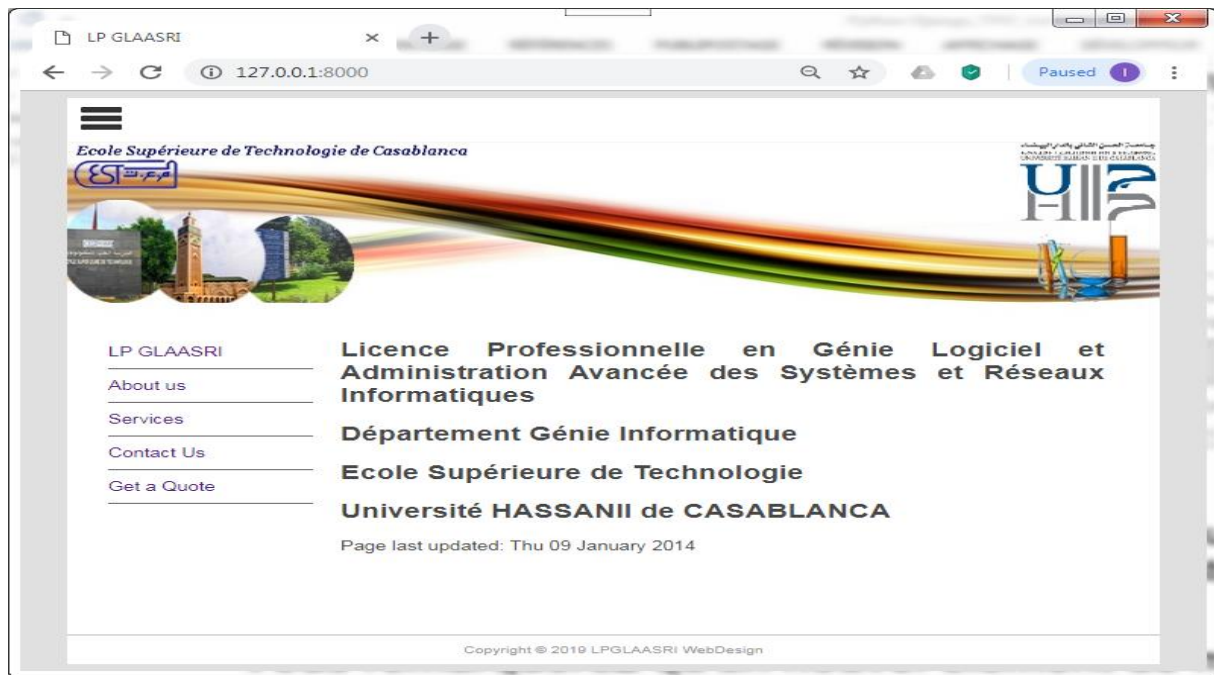
```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.quote_req, name='quote-request'),
]
```

Ajouter le lien au fichier base.html du projet comme indiqué ci-dessous.

```
# mysite\templates\base.html
# ...
{% block sidenav %}
<li>Menu 1</li><li>Menu 2</li><li>Menu 3</li>
{% endblock sidenav %}
<li><a href="/contact">Contact Us</a></li>
<li><a href="/quote/">Get a Quote</a></li>
```

Si vous accédez à <http://127.0.0.1:8000>, vous devriez voir une page semblable à celle de la figure ci-dessous.



Vous remarquerez qu'un nouvel élément de menu a également été ajouté au menu de gauche. Si vous cliquez sur le lien **Get a quote** vous obtenez le formulaire ci-dessous.

The screenshot shows a web browser window with the title 'Quote Request'. The address bar displays '127.0.0.1:8000/quote/'. The page features a header with the school's name in French and Arabic, a logo, and a decorative banner. A left sidebar contains navigation links: 'About us', 'Services', 'Contact Us', and 'Get a Quote'. The main content area is titled 'Quote Request' and contains a form with the following fields:

- Name:*
- Position:
- Company:
- Address:
- Phone:
- Email:*
- Web:
- Description:*
- Sitestatus:*
- Priority:*
- Jobfile: (with a 'Choose File' button and 'No file chosen' text)
- Submit button

At the bottom of the page, a copyright notice reads: 'Copyright © 2019 LPGLAASRI WebDesign'.

Entrez quelques informations maintenant, nous les utiliserons dans le TP suivant. N'oubliez pas de joindre des fichiers pour tester la capacité de téléchargement de fichiers de votre formulaire.