Django

1. Verifier si django est installé

py -m django --version

1. Créer un projet

django-admin startproject siteweb

1. Vérifier que le projet fonctionne : lancer le serveur de développement

py manage.py runserver

1. Créer une application

py manage.py startapp appliweb

LES FONCTIONS : fichier **views.py (V) 2°**

*la vue*

1. Définir une vue (la fonction de la page, son utilité : son nom)

Ouvrez le fichier **appliweb/views.py** et placez-y le code suivant

**from** **django.http** **import** render

**def** index(request):

**return** render(request, "appliweb/index.html")

1. Lier cette vue(s) à l’URL

LES URLs : fichier **urls.py 3°**

Dans le répertoire **appliweb**, créez un fichier nommé **urls.py**.

Dans le fichier **appliweb/urls.py**, insérez le code suivant :

**from** **django.urls** **import** path

**from** **.** **import** views

*# ajouter un espace de nom à l’application*

nom\_appli = 'appliweb'

urlpatterns = [

path('', index, name='index'),

path('connexion/', connexion, name='connexion'),

]

1. Dans **siteweb/urls.py**, inclure les URLs de l’application :  fonction **[include()](https://docs.djangoproject.com/fr/4.0/ref/urls/" \l "django.urls.include" \o "django.urls.include)**

**from** **django.contrib** **import** admin

**from** **django.urls** **import** include, path

urlpatterns = [

path('appliweb/', include('appliweb.urls')),

path('admin/', admin.site.urls),

]

1. Vérifier que la vue fonctionne

py manage.py runserver

1. Éditer le fichier **settings.py**

RÉGLAGES : fichier **settings.py**

Ouvrez **siteweb/settings.py**.

Définissez [**TIME\_ZONE**](https://docs.djangoproject.com/fr/4.0/ref/settings/#std-setting-TIME_ZONE) selon votre fuseau horaire.

Inscrire le nom de l’application dans [**INSTALLED\_APPS**](https://docs.djangoproject.com/fr/4.0/ref/settings/#std-setting-INSTALLED_APPS)

1. Configurer la base de données : lancer les migrations

py manage.py migrate

1. Activer les tables (les modèles)

Modifiez le fichier **mysite/settings.py** et inclure l’application au réglage [**INSTALLED\_APPS**](https://docs.djangoproject.com/fr/4.0/ref/settings/#std-setting-INSTALLED_APPS).

1. Définir la langue du site  [**LANGUAGE\_CODE**](https://docs.djangoproject.com/fr/4.0/ref/settings/#std-setting-LANGUAGE_CODE)
2. Dans le fichier  **mysite/settings.py**  éditer la variable DIRS

TEMPLATES = [{

'DIRS': [BASE\_DIR / 'templates'],

}]

1. xxx

LES TABLES : fichier **models.py (M) 1°**

*Les modèles*

1. Créer des tables (des modèles, le schéma de base de données) dans **appliweb/models.py**

**from** **django.db** **import** models

**class** **Table**(models.Model):

variable = type de variable(paramètres)

champ = type de champ(attributs)

colonne = type de colonne(paramètres)

1. Stocker ces tables sous forme de migration. ou appliquer leurs des modifications éventuellement effectuées :

py manage.py makemigrations appliweb

1. Créer ces tables dans la base de données

py manage.py migrate

1. Afficher ces tables (modèles) dans l’interface d’administration
2. xxxx

L’INTERFACE GRAPHIQUE : Le shell

1. Jouer avec l’interface de programmation (API) : utiliser un shell interactif Python pour jouer avec l’API

* Lancer un shell python

py manage.py shell

* Une fois dans le shell, explorez l’[API de base de données](https://docs.djangoproject.com/fr/4.0/topics/db/queries/):

LE SITE D’ADMINISTRATION

1. créer administrateur

py manage.py createsuperuser

1. redémarrer le serveur

py manage.py runserver

et aller à l’URL « /admin/ » de votre domaine local pour vous connecter avec votre compte

1. Il faut indiquer à l’admin que les objets (classes), par exemple : **Question** ont une interface d’administration

ouvrez le fichier **polls/admin.py** et éditez-le de la manière suivante :

LE TEMPLATES : fichiers **html (T)**

1. Créer un dossier nommé **templates** dans le répertoire **appliweb** dans lequel vous placez un nouveau fichier **index.html**.

Autrement dit :  **appliweb/templates/appliweb/index.html**

1. Insérer du code, des balises de gabarit
2. aaa

L’INTERFACE D’ADMINISTRATION :

fichier **admin.py 5°**

1. Afficher ou inscrire les tables (les modèles) dans l’interface d’administration

**from** **django.contrib** **import** admin

**from** **.models** **import** nomTable, nomClass

admin.site.register(nomTable)  *c-à-d nomClass*

1. Personnaliser le formulaire d’administration
2. Personnaliser l’apparence de l’interface d’administration
3. Personnaliser la page d’accueil de l’interface d’administration

**class** **QuestionAdmin**(admin.ModelAdmin):

champs = ['date\_publication', 'texte\_question']

admin.site.register(Question, QuestionAdmin)

1. FFFFFFFFFFF

LE FORMULAIRE

fichiers **forms.py 4°**

1. Créer un dossier nommé **templates** dans le répertoire **appliweb** dans lequel vous placez un nouveau fichier **index.html**.

LES TESTS

1. Créer un dossier nommé **templates** dans le répertoire **appliweb** dans lequel vous placez un nouveau fichier **index.html**.

MISE EN FORME, PERSONNALISATION

Fichiers statiques : **CSS, JavaScript...**

1. Créer un dossier nommé **static** à l’intérieur du dossier **appliweb**. À l’intérieur du dossier **static**, créer un autre dossier du même nom qu’ **appliweb** et placez-y un fichier **style.css**.

**Infos d’authentification : HttpRequest.user**

Provenant de **[AuthenticationMiddleware](https://docs.djangoproject.com/fr/4.0/ref/middleware/" \l "django.contrib.auth.middleware.AuthenticationMiddleware" \o "django.contrib.auth.middleware.AuthenticationMiddleware)**: une instance de [**AUTH\_USER\_MODEL**](https://docs.djangoproject.com/fr/4.0/ref/settings/#std-setting-AUTH_USER_MODEL) représentant l’utilisateur actuellement connecté. Si aucun utilisateur n’est actuellement connecté, **user** contiendra une instance de **[AnonymousUser](https://docs.djangoproject.com/fr/4.0/ref/contrib/auth/" \l "django.contrib.auth.models.AnonymousUser" \o "django.contrib.auth.models.AnonymousUser)**. Vous pouvez les différencier par l’attribut **[is\_authenticated](https://docs.djangoproject.com/fr/4.0/ref/contrib/auth/" \l "django.contrib.auth.models.User.is_authenticated" \o "django.contrib.auth.models.User.is_authenticated)**, comme ceci :

**if** request.user.is\_authenticated:

... *# Do something for logged-in users.*

**else**:

... *# Do something for anonymous users.*