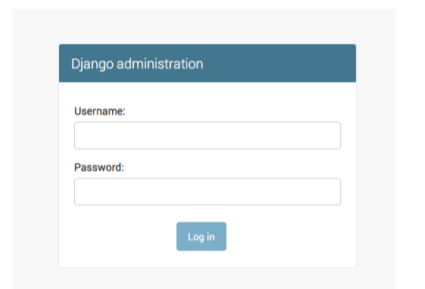
DJANGO: Edukia

1. **Modeloak Djangon**
   1. Datu basearen mapeoa objetuetara
2. **Django kudeaketa (admin) arloa**
   1. **admin.py** fitxategian modeloa inportatu eta erregistratu

****

* 1. **Python manage.py runserver** ejekutatu eta nabigatzailean 

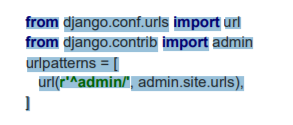
****

* 1. Supererabiltzailea sortu

**python manage.py createsuperuser**

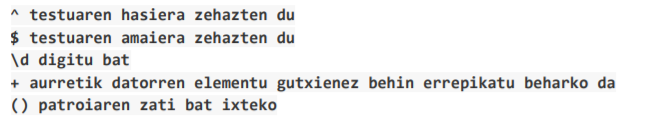
1. **Django urlak**

**Urls.py**

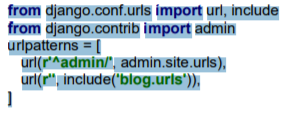


Regex bidez, Django bistak URL helbideekin bat egiten dute. Regex (expresio erregularrak) bilaketa patroia sortzen duten arau asko (pila bat) dauzka.

Oinarrizko patroiak nola sortzen diren azaltzeko, honako ezaugarriak kontutan hartuko ditugu:



Proiektuaren fitxategia garbi mantentzea da gure helburua. Hori lortzeko, gure aplikazioaren URL-ak inportatuko ditugu “proiektuaren/urls.py” fitxategi nagusira.



* **Aplikazioko urls.py**



***\*\* Momentuan errorea, views.py fitxategian metodoa oraindik ez dugulako***

1. **Django bistak**

“View” bat da gure aplikazioaren logika zehazten dugun kokapena. Aurretik sortutako “model”-aren informazioa eskatuko da eta sortuko den “view” batera pasatuko da, hori da helburua, informazioa bistaratzea. Bistak (view-ak) Python-eko metodoak dira.

* **“aplikazioa/view.py”**

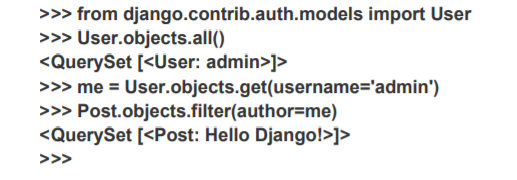
****

“render” funtzioaren barruan “request” parametroa daukagu (Erabiltzailearengatik jasotako guztia Internet bidez) eta “blog/post\_list.html” fitxategia plantilla bezala.

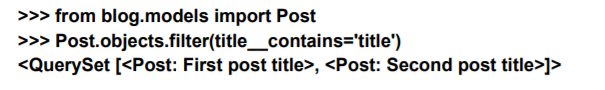
1. **Django ORM eta QuerySet-ak.**
   * **Objektuak filtratu.**

QuerySet arloko punturik garrantzitsuena da edukia filtratu ahal izatearena. Horretarako **“filter”** erabiliko dugu **“all”** izan beharrean **“Post.objects.all()”** eskaeran.

Adibidez**:**

****

Sortuta ditugun post guztiak kontutan hartuta, post-aren “title” hitza duen post guztiak lortu nahiko dira:

****

**Oharra** *Bi azpi gidoi (\_) daude title eta contains artean. Django ORM-ak sintaxi hau erabiltzen du eremuetako izenak banandu ahal izateko (“title”) eta eragiketa edo filtroak (“contains”). Azpi gidoi bat erabiliz gero, "****FieldError: Cannot resolve keyword title\_contains****" bezalako errorea jasoko dugu.*

* + **Objektuak ordenatu.**

QuerySet-ak objektu zerrenda ordenatzea ahalbidetzen dizu. Adibidez, “created\_date” eremuarengatik ordenatuko dira adibide gisa:



Alderantziz ordenatu nahi bada, “created\_date” aurretik ‘-’ jarriko dugu:

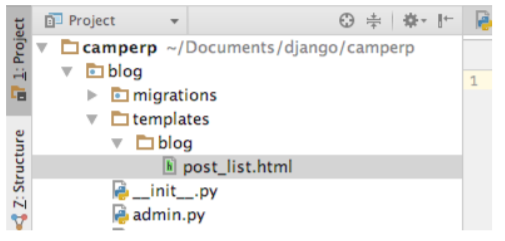
****

1. **Django templateak**

Plantilla bat sortzeko (blog aplikazioan adibidez) aplikazio bakoitzaren barruan plantillak gorde ahal izateko.

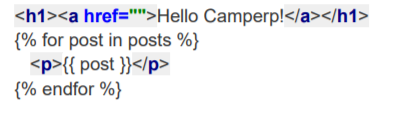
*Berrabiarazi zerbitzaria.*

HTML-ean informazioa ematea ahalbidetzen digu “Django template tags”-ek, honek webguneak eduki dinamikoa eta errazago kudeatzeko aukera ematen digu.

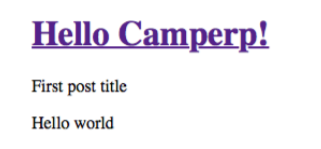
****

****

**HTMLan:**

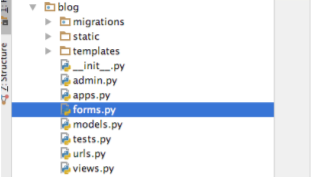
****

**Nabigatzailean:**

****

1. **Django formularioak**

“model” (models.py), “view” (views.py) eta horrelako Django zati garrantzitsuenetan bezala, “forms” bere fitxategi propioa izango du “**forms.py**” izenekoa.

****

Fitxategi honen kasuan, gutxienez kode hau beharko du:

****

PostForm formularioaren barruan “Meta” izeneko klase bat daukagu, non zer modelo erabili beharko dugun zehaztu beharko zaio, kasu honetan Post modeloa izanda, honela:

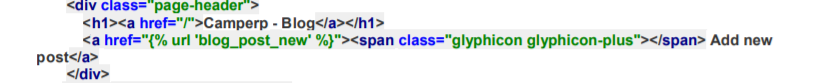
model = Post.

Gero zer eremu(ak) gehitu nahi ditugun gure formularioan zehaztuko dugu.

Formularioa “view” baten erabiltzeko behar ditugun pausoak burutu beharko ditugu gure plantillan erakutsi ahal izateko. “view” berri honetara sarbidea izateko esteka bidez lehen egindako pausoak bete behar ditugu:

**● Sortu orrialdera esteka bat.**

Formulario batera egiteko lotura, hasierako “**base**” plantillan egingo dugu eta honi esteka gehitzeko, “**blog/templates/blog/base.html**” zabalduko dugu eta esteka bat gehituko dugu “page-header” izeneko “div” batean:

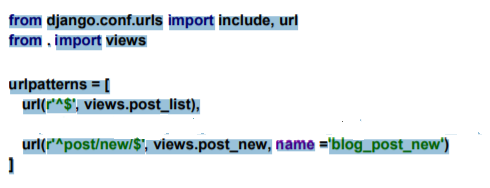


*\*\*\*\* Honelako errorea emango digu URL-a definitzen ez dugun bitartean:*

**

**● URL helbide bat zehaztu.**

Aurreko puntuan emandako errorea konpontzeko, honako URL-a gehituko diogu “blog” *aplikazio* barruko “urls.py” fitxategian:



***\*\*\*\**** *Berriro saiatzen bagara bistaratzen hasierako orrialdea “post\_new” bista ez dagoenez inplementatua “AttributeError” emango digu.*

*“AttributeError: 'module' object has no attribute 'post\_new'”*

**● Bista (view) bat zehaztu.**



Orain BAI ikusiko da zuzen gure hasierako orrialdea.

**● Plantilla HTML bat zehaztu**

Estekaren arloa eta bere bista sortu ondoren, estekak eramango gaituen leihorako plantilla definitzea falta zaigu.

“post\_edit.html” fitxategia izango da sortuko den fitxategia helburu honetarako eta hau “blog/templates/blog” direktorioan gordeko dugu.

Formulario batek funtzionatu dezan, zenbait puntu kontutan eduki beharko ditugu:

● Formularioa erakutsi behar dugu eta hau egiteko **{{form.as\_p}}** batekin nahikoa izango dugu.

● Aipatutako aurreko kodea HTML kodearen barruan egon beharko litzateke:

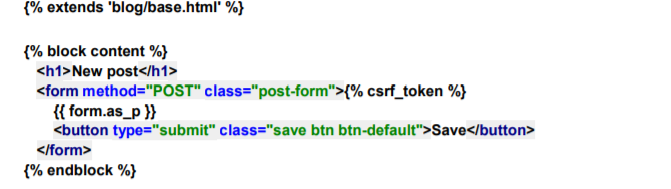
“<form> {{form.as\_p}}</form>”

● Informazioa gordetzeko botoi baten beharra daukagu. HTMLan botoia gehituko dugu.

“<button type=’submit’ > Gorde </button>

● <form> kodearen ostean {% csrf\_token %} gehitzea gomendatzen da. Ez da derrigorrezkoa BAINA zure formularioak SEGURUAGOAK izatea ahalbidetzen du.

Kodea honela erakutsiko da:



* **Formularioa gorde.**

Zabaldu “views.py” fitxategia eta “post\_new” funtzioan, datuak gordetzeko prestatu behar dugu. Guri interesatzen zaiguna da informazioa hartzea, gordetzea eta gehitutakoa ikusi ahal izatea. Hau egiteko, definitutako “POST” metodoan oinarrituko gara formularioan zehaztu dena tratatu eta gorde ahal izateko.

|  |
| --- |
|  |



