Amsterdam UMC
Universitair Medische Centra

Apotheek

Django 3.0 Cheat Sheet (Dutch)

- By: Dion Dresschers of Amsterdam UMC

- Version: 2020-03-18 08:02:32

(Optioneel) Check Ubuntu versie:

```
dion@desktop:~/data/git/django/v/lifeplanning-project$ cat /etc/*release* | grep
VERSION=
VERSION="18.04.4 LTS (Bionic Beaver)"
```

(Optioneel) Maak een Virtual ENVironment

```
dion@desktop:~/data/git/django$ python3 -m venv v
```

(Optioneel) Activeer die Virtual ENVironemnt

```
dion@desktop:~/data/git/django$ cd v
dion@desktop:~/data/git/django/v$ source bin/activate
(v) dion@desktop:~/data/git/django/v$
```

Installeer Django (versie 3.0.4)

```
(v) dion@desktop:~/data/git/django/v$ pip install Django==3.0.4
```

Maak een djangoproject genaamd: 'lifeplanning'

```
(v) dion@desktop:~/data/git/django/v$ django-admin startproject <lifeplanning>
```

(Optioneel) Hernoem de directory 'lifeplanning' naar 'lifeplanning-project'

```
(v) dion@desktop:~/data/git/django/v$ mv lifeplanning/ lifeplanning-project
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$
(v) dion@desktop:~/data/git/django/v$ cd lifeplanning-project/
```

(Optioneel) Bekijk de files in het project 'lifeplanning-project'

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ tree
.
├── lifeplanning
│   ├── asgi.py
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py

1 directory, 6 files
```

Host de (test)server op je locale computer

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ python3 manage.py
runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you
apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

March 18, 2020 - 07:11:59
Django version 3.0.4, using settings 'lifeplanning.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

(Optioneel) Test of de server draait in de internetbrowser Firefox (alleen voor Ubuntu met een GUI)

```
dion@desktop:~$ firefox --new-window http://127.0.0.1:8000 &
```

Maak een nieuw app aan genaamd 'todo'

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ python3 manage.py
startapp todo
```

(Optioneel) Bekijk de files in het project 'lifeplanning-project'

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ tree
.
├── lifeplanning
│   ├── asgi.py
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   └── settings.cpython-36.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── manage.py
└── todo
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
    │   └── __init__.py
    ├── models.py
    ├── tests.py
    └── views.py

4 directories, 15 files
```

Voeg de nieuw app genaam 'todo' toe aan de `lifeplanning/settings.py` tuple

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'todo',
]
```

Controleer of er databaseveranderingen zijn

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ python3 manage.py
makemigrations
No changes detected
```

Migreer databaseveranderingen

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ python3 manage.py
migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$
```

Voeg een URL toe via de `portfolio/urls.py`

```
from todo import views # Dit importeert de file `todo/views.py`, nodig voor de
functie 'signupuser' die nog gaan aanmaken.

urlpatterns = [
    # ...
    path('signup/', views.signupuser, name="signupuser"),
]
```

In de `todo/views.py` file voeg de functie 'signupuser' toe

```
def signupuser(request):
    return render(request, 'todo/signupuser.html')
```

Maak de HTML/Twig-template aan

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ mkdir -p
todo/templates/todo
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ touch
todo/templates/todo/signupuser.html
```

Geef text in de HTML/Twig-file `todo/templates/todo/signupuser.html` zodat je weet dat de View werkt.

```
Je bent uitgekomen op 'singupuser'.
```

(Optioneel) Test of de View werkt.

```
dion@desktop:~$ firefox --new-window http://127.0.0.1:8000/signup &
```

Maak gebruik van een Django Formulier

In de 'todo/views.py'-file, importeer de UserCreationFrom

```
from django.contrib.auth.forms import UserCreationForm
```

Geef het mee als attribuut aan de 'render'-functie, waarbij de 'form' de variabele is voor de Twig-template.

```
def signupuser(request):
    return render(request, "todo/signupuser.html", {'form':UserCreationForm()})
```

In de Twig-template file `todo/templates/todo/signupuser.html` gebruik de eerder doorgestuurde variabele.

```
{{ form }}
```

Echter om het formulier succesvol te gebruiken, zet het tussen `<form>`-tags, plaats een submit-button en maak gebruik van een CSRF-token:

```
```

```
<form method="POST">
{% csrf_token %}
{{ formulier }}
<button type="submit">Verstuur</button>
</form>
```

Check if a 'GET', or 'POST'-method is used in de `todo/views.py'-file.

```
def signupuser(request):
    if request.method == "POST":
        # Het is een post
    if request.method == "GET":
        return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm()}) # Laat weer het formulier zien
```

Maak een gebruiker aan.

In de `todo/views.py`-file:

```
from django.contrib.auth.models import User
```

```
def signupuser(request):
    if request.method == "GET":
        return render(request, "todo/signupuser.html",
{'form':UserCreationForm()}) # Laat weer het formulier zien
    if request.method == "POST":
        if request.POST['password1'] == request.POST['password2']:
            user = User.objects.create_user(request.POST["username"],
password=request.POST["password1"])
            user.save()
    else:
        temp = 0;
        # Tell the user the passwords do not match
```

Maak een admin User aan:

```
(v) dion@desktop:~/data/git/django/v/lifeplanning-project$ python3 manage.py
createsuperuser
```

Check op basis van wachtwoorden die niet overeenkomen

In de `todo/views.py`-file:

```
def signupuser(request):
    if request.method == "GET":
        return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm()})
    if request.method == "POST":
        if request.POST['password1'] == request.POST['password2']:
            user = User.objects.create_user(request.POST["username"],
password=request.POST["password1"])
            user.save()
        else:
            return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm(), 'foutmelding':"De wachtwoorden komen niet
overeen."})
```

In de `templates\totod\signupuser.html`-file.

Boven de `<form>`-tag:

```
<p> {{ foutmelding }} </p>
```

Check op basis van dat de gebruiker al bestaat.

In de `todo/views.py`-file:

```
from django.db import IntegrityError
```

Gebruik een `try` en `catch` en maak gebruik van de `IntegrityError`

```
def signupuser(request):
    if request.method == "GET":
        return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm()})
    if request.method == "POST":
        if request.POST['password1'] == request.POST['password2']:
            try:
                user = User.objects.create_user(request.POST["username"],
password=request.POST["password1"])
                user.save()
            except IntegrityError:
                return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm(), 'foutmelding':"De gebruiker bestaat al."})
        else:
            return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm(), 'foutmelding':"De wachtwoorden komen niet
overeen."})
```

Laat de gebruiker inloggen met de functie: login() en redirect de user

```
from django.contrib.auth import login
from django.shortcuts import redirect
```

```
def signupuser(request):
    if request.method == "GET":
        return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm()})
    if request.method == "POST":
        if request.POST['password1'] == request.POST['password2']:
            try:
                user = User.objects.create_user(request.POST["username"],
password=request.POST["password1"])
                user.save()
                login(request, user) # log de gebruiker in
                return redirect('todos') # redirect de gebruiker
            except IntegrityError:
                return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm(), 'foutmelding':"De gebruiker bestaat al."})
        else:
            return render(request, "todo/signupuser.html",
{'formulier':UserCreationForm(), 'foutmelding':"De wachtwoorden komen niet
overeen."})
```

Jinja 2 Create a base file

```
lifeplanning-project$ touch todo/templates/todo/base.html
```

In die `base.html`-file maak gebruik van `block`, hier komt de informatie die pagina specifiek voor de andere HTML pagina is.

```
Deze informatie komt in elke file die {% extends 'todo/base.html' %} gebruikt.

Onderstaande info verschilt per andere pagina.
{% block %}
{% endblock %}
```

In de andere file (bijvoorbeeld 'todo/templates/todo/todos.html), kan je de `base.html` overnemen met `extends`.

```
{% extends 'todo/base.html' %}

{% block inhoud %}
    Dit is specifieke info alleen voor de pagina Todos
{% endblock %}
```

Check of een gebruiker ingelogd is in Jinja 2

Zie dat onderstaande A-links nog niks doen.

```
{% if user.is_authenticated %}
    Je bent ingelogd als {{ user.username }}
    <!-- optionally show a Logout-button -->
{% else %}
    <!-- show a Sign Up-button -->
    <!-- show a Login-button -->
{% endif %}
```

Maak een Logout-button

Maak eerst een nieuwe entry in de `urlpatterns` in de `urls/py`-file.

```
urlpatterns = [
    # ...
    path('logout/', views.logoutuser, name="logoutuser"),
]
```

Dan maak je de functie aan in de 'todo/views.py':

```
def logoutuser(request):
    if request.method == "POST": # Belangrijk, want sommige browers gaan
automatische alle buttons na met 'GET' om data te harversten.
        logout(request)
        return redirect('home') # De 'home' is nog niet aangemaakt.
```

Dan maak je in de HTML-template In de HTML-template gebruik geen , want dat is een "GET" en sommige browsers proberen alle alvast binnen te snoepen.

```
<!-- dit dus NIET doen: <a href="{% 'logoutuser' %}"> -->
```

Maar gebruik een 'POST', de manier om dit te doen is om eem formuliertje te maken met de method "POST":

```
    <form action="{% url 'logoutuser' %}" method="POST">
        {% csrf_token %}
        <button type="submit">Loguit</button>
    </form>
```

Laat de gebruiker inloggen

Maak eerst een nieuwe entry in de `urlpatterns` in de `urls/py`-file.

```
urlpatterns = [
    # ...
    path('login/', views.loginuser, name="loginuser"),
]
```

Dan maak je de functie aan in de 'todo/views.py':

```
from django.contrib.auth.forms import AuthenticationForm
```

```
def login(request):


```

Dan maak je de HTML-template aan.

```
$ touch todo/templates/todo/loginuser.html
```

```
{% extends 'todo/base.html' %}

{% block inhoud %}
<p>
Je bent uitgekomen op 'loginuser'.
</p>

{{ foutmelding }}
<form method="POST">
    {% csrf_token %}
    {{ formulier }}
    <button type="submit">Login</button>
</form>
{% endblock %}
```

Maak een button aan om in te loggen in de `base.html`:

```
    <a href="{% url 'signupuser' %}">Signup</a>
    <a href="{% url 'loginuser' %}">Login</a>
```

Zorg dat de gebruiker daadwerkelijk in de database staat. In `views.py`

```
from django.contrib.auth import authenticate
```

```
def loginuser(request):
    if request.method == "GET":
        return render(request, "todo/loginuser.html",
{'formulier':AuthenticationForm()})
    else:
        user = authenticate(request, username=request.POST['username'],
password=request.POST['password']) # Als het authenticeren faalt, dan is het
resultaat: None
        if user is None:
            return render(request, "todo/loginuser.html",
{'formulier':AuthenticationForm(), 'foutmelding':'Het authenticeren is niet
gelukt.'})
        else: #
            login(request, user)
            return redirect("currenttodos")
```

Make a Model (database entries)

In de models.py file:

```
from django.db import models # deze is er al
from django.contrib.auth.models import User # deze als we gebruik willen maken van
User


class todo(models.Model):
    titel = models.CharField(max_length=100)
    tekst = models.TextField(blank=True) # Optionally to fill in, can leave blank,
can be Null, can be empty
    tijdstip_start = models.DateTimeField(auto_now_add=True) # Automatisch
aangemaakt
    prio = models.CharField(max_length=1)
    # klaar = models.BooleanField()
    tijdstip_klaar = models.DateTimeField(null=True, blank=True) # null=True moet
toegevoegd zijn, in verband met complexe datum & tijd
    todo_user = models.ForeignKey(User, on_delete=models.CASCADE) # Making use of
Foreign key, and import the User
```

Maak de migraties klaar om te migeren naar de database

```
dhdresschers@19-001596:/mnt/c/Users/dhdresschers/data/git/django/v/lifeplanning-
project$ python3 manage.py makemigrations
Migrations for 'todo':
  todo/migrations/0001_initial
```

Migreer naar de database

```
dhdresschers@19-001596:/mnt/c/Users/dhdresschers/data/git/django/v/lifeplanning-
project$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, todo
Running migrations:
  Applying todo.0001_initial... OK
```

Je moet je Model nog importeren in de Admin pagina om hem te kunnen zien

Je zal zien dat het nieuwe model nog niet zichtbaar is in de /admin pagina:

dion@desktop:~$ firefox --new-window http://127.0.0.1:8000/admin &

In `admin.py`

```
from django.contrib import admin # Deze was er al
from .models import todo

# Register your models here.

admin.site.register(todo)
```

dion@desktop:~$ firefox --new-window http://127.0.0.1:8000/admin &

Maak de todo-object ook descriptive in de /admin pagina.

In de `models.py`-pagina

```
class todo(models.Model):

    def __str__(self):
        return self.title # nodig, anders worden de titels in de /admin pagina
```

```
iets van 'todo object (1)' en 'todo object (2)' in plaats van 'Fietsen' en
'Koffieleuten met Jannie'
```

Voeg het automatische ingevulde veld 'tijdstip_start' toe aan de /admin pagina

In `models.py` staat:

```
class todo(models.Model):
    # ...
    tijdstip_start = models.DateTimeField(auto_now_add=True) # Automatisch
aangemaakt
```

Dit veld wordt automatisch ingevuld en wordt standaard niet weergegeven in de /admin.

In de `admin.py`-file:

```
class LaatHiddenZien(admin.ModelAdmin):
    readonly_fields = ('tijdstip_start',)

# Register your models here.

admin.site.register(todo, LaatHiddenZien)
```

Maak een formulier aan waar personen data kunnen invoeren

Maak in `ursl.py` een nieuwe entry aan in 'urlpatterns=[]'

urlpatterns = [# ... path('create/', views.createtodo, name="createtodo"),]

Maak een nieuwe functie aan in 'todo/view.py'

```
from .forms import TodoFrom # de .forms is de zelf aangemaakte `todo\forms.py`-
file in het project
```

```
def createtodo(request):
    if request.method == 'GET': # iemand wilt data ophalen
        return render(request, 'todo/createtodo.html',
{'formulier':MaakTodoFormulier})
    else:
        pass
```

Maak een nieuwe template aan

```
(v)$ touch todo\templates\todo\createtodo.html
```

```
{% extends 'todo/base.html' %}

{% block inhoud %}
<h1>Maak todo</h1>
{{ foutmelding }}
<form method="POST">
    {% csrf_token %}
    {{ formulier }}
    <button type="submit">Opslaan</button>
</form>
{% endblock %}
```

Maak een nieuwe file aan 'todo/forms.py'

```
from django.forms import ModelForm
from .models import todo # dit is de `models.py` file in het project
```

```
def createtodo(request):
    if request.method == 'GET': # iemand gebruikt GET, dus via URL, dus drukt een
knop, dus laat formulier zien
        return render(request, 'todo/createtodo.html',
{'formulier':MaakTodoFormulier()})
    else: # iemand gebruit 'POST', dus gebruikt formulier dus zend data naar de
database/model
        formulier = MaakTodoFormulier(request.POST) #
        nieuwetodo = formulier.save(commit=False) # commit=False, is maak een
nieuw Object, maar zat het nog niet in de database, zonder dit het zou het meteen
opslaan
        nieuwetodo.todo_user = request.user # pas het object aan zodat er ook de
'user' in zit.
        nieuwetodo.save()
        return redirect("currenttodos")
```

Doe een try-except check op het moment dat iemand het HTML-formulier aanpast en zo incorrecte data probeert te pushen

```
def createtodo(request):
    if request.method == 'GET': # iemand gebruikt GET, dus via URL, dus drukt een
knop, dus laat formulier zien
        return render(request, 'todo/createtodo.html',
{'formulier':MaakTodoFormulier()})
    else: # iemand gebruit 'POST', dus gebruikt formulier dus zend data naar de
database/model
```

```
        try:
            # dit voorkomt django errors als bv data niet goed is ingevoerd, wat
initieel opgevangen wordt door het html formulier
            formulier = MaakTodoFormulier(request.POST) #
            nieuwetodo = formulier.save(commit=False) # commit=False, is maak een
nieuw Object, maar zat het nog niet in de database, zonder dit het zou het meteen
opslaan
            nieuwetodo.todo_user = request.user # pas het object aan zodat er ook
de 'user' in zit.
            nieuwetodo.save()
            return render('todo/currenttodo.html')
        except ValueError:
            return render(request, 'todo/createtodo.html',
{'formulier':MaakTodoFormulier(), 'foutmelding':'Je probeert rare data te
pushen'})
```

Data weergeven vanuit de database

In views.py

```
from .models import todo
```

```
def currenttodos(request):
    todos = todo.ojects.all() # echter haalt het de todos van alle gebruikers op
ipv alleen specifieke gebruiker
    return render(request, "todo/currenttodos.html", {'todos':todos})
```

Laat nu alleen de todos van de gebruiker zelf zien:

```
def currenttodos(request):
    todos = todo.objects.filter(todo_user=request.user)
    return render(request, "todo/currenttodos.html", {'todos':todos})
```

In de template `todo/currenttodos.html`:

```
    <ul>
        {% for todo in todos %} <!-- todo is zelfgekozen, todos is doorgestuurd
vanuit de functie -->
        <li>{{ todo.titel }}</li>
        {% endfor %}
    </ul>
```

Geef niet de data weer van zaken die een waarde hebben (waar een waarde niet null is)

```
def currenttodos(request):
    todos = todo.objects.filter(todo_user=request.user,
tijdstip_klaar__isnull=True)
    return render(request, "todo/currenttodos.html", {'todos':todos})
```

Maak de resultaten vet op basis van een bepaalde waarde

In de template:

```
{% extends 'todo/base.html' %}

{% block inhoud %}
Dit zijn je todo's:

<ul>
    {% for todo in todos %}
        <li>
            {% if todo.prio == "1" %}<b>{% endif %}
                {{ todo.titel }}
            {% if todo.prio == "1" %}</b>{% endif %}
        </li>
    {% endfor %}
</ul>

{% endblock %}
```

Laat de inhoud zien als er een bepaald tekst veld is ingevuld

```
    {% for todo in todos %}
        <!-- ... -->
            {% if todo.tekst %} - {{ todo.tekst }} {% endif %}
        <!-- ... -->
    {% endfor %}
```

Laat een detailed pagina zien van een todo entry

In de `urls.py` maakl een nieuwe 'urlpattern' aan:

```
urlpatters = [
    # ...
    path('todo/<int:todo_primarykey'), views.viewdetail, name="viewdetail"),
]
```

In de `views.py` maak een nieuwe functie aan:

```
from django.shortcuts import get_object_or_404
```

```
def viewdetail(request, todo_primarykey):
    tododetail = get_object_or_404(todo, pk=todo_primarykey) # de tweede todo, is
de Class die we aangemaakt hebben in `models.py`
    return render(request, 'todo/viewdetail.html', {'tododetail':tododetail} )
```

Maak de template aan:

```
(v)$ touch todo/templates/todo/viewdetail.html
```

Vul de template in:

```
{% extends 'todo/base.html' %}

{% block inhoud %}
Detailpagina

{{ tododetail.titel }}

{% endblock %}
```

Pas de URL in de overzichtspagina zo aan dat er een link is naar de detailpagina

In `currenttodos.html':

```
    {% for todo in todos %}
        <li>
            <a href="{% url 'viewdetail' todo.id %}">
                <!-- ... -->
            </a>
        </li>
    {% endfor %}
```

Laat de detailpagina weergeven in een formulier (maar nog zonder 'Edit')

In de `views.py`:

voeg een formulier toe en vul deze met reeds aanwezige informatie uit de database/model:

```
def viewdetail(request, todo_primarykey):
    tododetail = get_object_or_404(todo, pk=todo_primarykey) # de tweede todo, is
de Class die we aangemaakt hebben in `models.py`
    formulier = MaakTodoFormulier(instance=tododetail)# deze is anders, dat hier
reeds data in staat
    return render(request, 'todo/viewdetail.html', {'tododetail':tododetail,
'formulier':formulier } )
```

In de `viewdetail.html`:

```
{% extends 'todo/base.html' %}

{% block inhoud %}
Detailpagina

{{ tododetail.titel }}

<form action="POST">
    {{ formulier }}
</form>

{% endblock %}
```

Zorg dat de data in het formulier klaargemaakt kan worden voor een Edit.

```
def viewdetail(request, todo_primarykey):
   tododetail = get_object_or_404(todo, pk=todo_primarykey) # de tweede todo, is
de Class die we aangemaakt hebben in `models.py`
    if request.method == "GET":
        formulier = MaakTodoFormulier(instance=tododetail)# deze is anders, dat
hier reeds data in staat
        return render(request, 'todo/viewdetail.html', {'tododetail':tododetail,
'formulier':formulier } )
    else:
        try:
            formulier = MaakTodoFormulier(request.POST, instance=tododetail) # De
instance is nodig om ook de User mee te geven... (anders error)
            formulier.save()
            return render(request, 'todo/viewdetail.html',
{'tododetail':tododetail, 'formulier':formulier } )
        except ValueError:
            return render(request, 'todo/viewdetail.html',
{'tododetail':tododetail, 'formulier':formulier, 'foutmeldingkje':'Geen goede
informatie' } )
```

In de `viewdetail.html` maak een 'submit'-button en 'csrf_token':

```
{% extends 'todo/base.html' %}

{% block inhoud %}
Detailpagina

{{ foutmeldingje }}
{{ tododetail.titel }}

<form method="POST">
    {% csrf_token %}
    {{ formulier }}
    <button type="submit">Edit/Save</button>
</form>

{% endblock %}
```

Iedereen kan echter alle todo_primarykey zien

```
def viewdetail(request, todo_primarykey):
 tododetail = get_object_or_404(todo, pk=todo_primarykey, todo_user=request.user)
```

Items markeren als 'klaar' of 'gearchifeerd'.

Maak een nieuwe functie aan (maar geen nieuwe template)

```
from django.utils import timezone
```

```
def completetodo(request, todo_primarykey):
    tododetail = get_object_or_404(todo, pk=todo_primarykey,
todo_user=request.user) # de tweede todo, is de Class die we aangemaakt hebben in
`models.py`
    if request.method == "POST":
        tododetail.tijdstip_klaar = timezone.now()
        tododetail.save()
        return redirect('currenttodos')
```

In de reeds bestaande 'viewdetail.html' maak een extra formulier aan met alleen de submit button.

```
{% extends 'todo/base.html' %}

{% block inhoud %}
Detailpagina
```

```
{{ foutmeldingje }}
{{ tododetail.titel }}

<form method="POST">
    {% csrf_token %}
    {{ formulier }}
    <button type="submit">Edit/Save</button>
</form>

<form method="POST" action="{% url 'completetodo' tododetail.id %}">
    {% csrf_token %}
    <!-- {{ formulier }} is weggehaald, anders is deze informatie dubbel -->
    <button type="submit">Complete/Archiveer</button>
</form>

{% endblock %}
```

Delete een item permanent van de model/database

In `urls.py`:

```
urlpatterns = [
    # ...
    path('todo/<int:todo_primarykey>/delete', views.deletetodo,
name="deletetodo"),
]
```

In `views.py:

```
def deletetodo(request, todo_primarykey):
    tododetail = get_object_or_404(todo, pk=todo_primarykey,
todo_user=request.user) # de tweede todo, is de Class die we aangemaakt hebben in
`models.py`
    if request.method == "POST":
        tododetail.delete()
        return redirect('currenttodos')
```

In `viewdetail.html` maak een nieuwe form aan met alleen een button:

```
{% extends 'todo/base.html' %}

{% block inhoud %}
Detailpagina

{{ foutmeldingje }}
{{ tododetail.titel }}
```

```
<form method="POST">
    {% csrf_token %}
    {{ formulier }}
    <button type="submit">Edit/Save</button>
</form>

<form method="POST" action="{% url 'completetodo' tododetail.id %}">
    {% csrf_token %}
    <!-- {{ formulier }} is weggehaald, anders is deze informatie dubbel -->
    <button type="submit">Complete/Archiveer</button>
</form>


<form method="POST" action="{% url 'deletetodo' tododetail.id %}">
    {% csrf_token %}
    <!-- {{ formulier }} is weggehaald, anders is deze informatie dubbel -->
    <button type="submit">Verwijder</button>
</form>

{% endblock %}
```

Laat completed items zien

In `urls.py`:

urlpatterns = [# ... path('klaar/', views.klaartodos, name="klaardtodos"),]

In `views.py`:

def klaartodos(request): klaartodos = todo.objects.filter(todo_user=request.user, tijdstip_klaar_isnull=False) return render(request, "todo/klaartodos.html", {'klaartodos':klaartodos})

Maak een template aan:

```
(v)$ touch todo/templates/todo/klaartodos.html
```

Maak de template aan:

```
{% extends 'todo/base.html' %}

{% block inhoud %}
De todos die klaar zijn...

<ul>
    {% for todo in klaartodos %}
        <li>
            <a href="{% url 'viewdetail' todo.id %}">
                    {{ todo.titel }}
                {% if todo.tekst %} - {{ todo.tekst}} {% endif %}
```

```
            </a>
                {{ todo.tijdstip_klaar|date:'M j Y H:i' }}

        </li>
    {% endfor %}
</ul>

{% endblock %}
```

Sorteer de items

```
def klaartodos(request):
    klaartodos = todo.objects.filter(todo_user=request.user,
tijdstip_klaar__isnull=False).order_by('-tijdstip_klaar') # Het min-teken '-' is
optioneel en inverteert dus
    return render(request, "todo/klaartodos.html", {'klaartodos':klaartodos})
```

In de base.html voeg links to naar alle mogelijke links

```
{% if user.is_authenticated %}
    Je bent ingelogd als {{ user.username }}

<a href="{% url 'createtodo' %}">Create</a>
<a href="{% url 'currenttodos' %}">Alle todos</a>
<a href="{% url 'klaartodos' %}">Afgeronde todos</a>

<!-- ... -->
```

Gebruikers de login_required() functie

Gebruikers die nog niet ingelogd zijn, kunnen nog teveel pagina's zien.

In `views.py`

```
from django.contrib.auth.decorators import login_required
```

Dan boven elke 'view' in `views.py` die gebruikers alleen mogen zien na inloggen '@login_required'

dus:

```
@login_required
def (request):
    todos = todo.objects.filter(todo_user=request.user,
tijdstip_klaar__isnull=True)
    return render(request, "todo/currenttodos.html", {'todos':todos})
```

```
@login_required
def currenttodos(request):
    todos = todo.objects.filter(todo_user=request.user,
tijdstip_klaar__isnull=True)
    return render(request, "todo/currenttodos.html", {'todos':todos})
```

Maar:

```
def signupuser(request):
    if request.method == "GET":
        # ...
```

We krijgen dan een '404' foutmelding.

Om te zorgen dat gebruikers niet de '404' melding krijgen, maar geredirect worden naar de login pagina:

In de `settings.py` voeg toe:

```
LOGIN_URL = "/login"
```

De "/login" refereert naar de reedsbestaande 'urlpatterns', waar we de login pagina gemaakt hebben:

```
    path('login/', views.loginuser, name="loginuser"),
```