Lab: Forms

1. Fill out the Form!

Create an application that displays the name, age, email, password and text from a user's input. HTML will be provided.

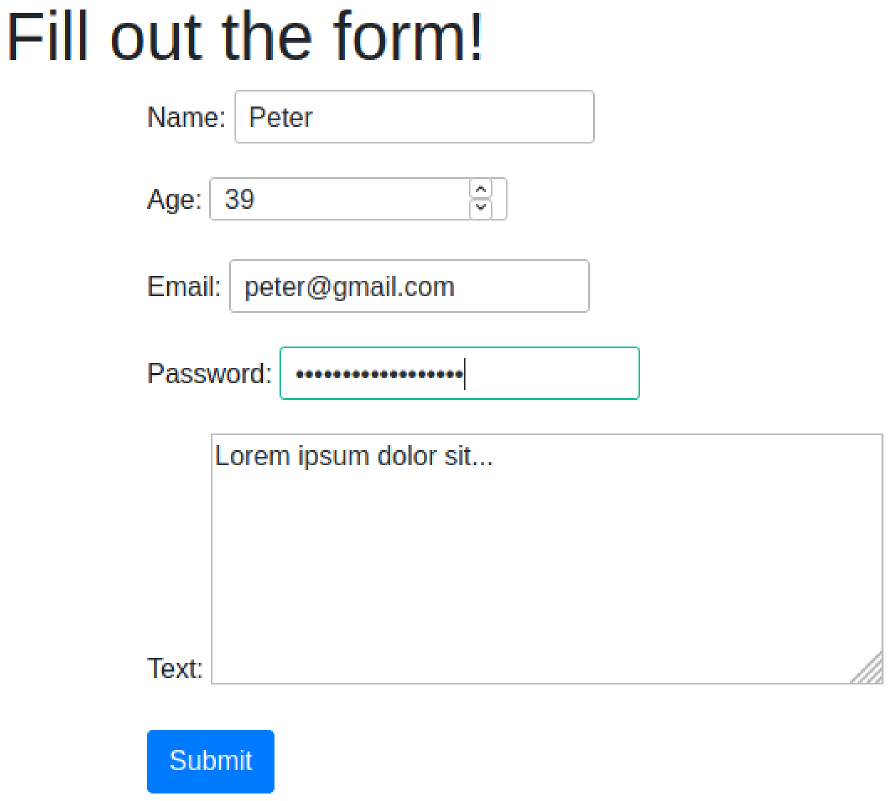
`Name: CharField`

`Age: IntegerField with NumberInput widget`

`Email: EmailField with EmailInput widget`

`Password: CharField with PasswordInput widget`

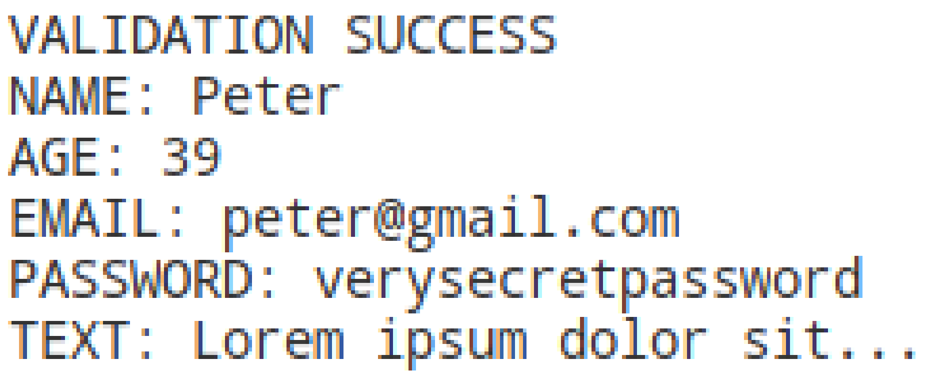
`Text: CharField with Textarea widget`



Fill out the form!

Name:	Peter
Age:	39
Email:	peter@gmail.com
Password:	••••••••••••••••••••
Text:	Lorem ipsum dolor sit...

The output on the console should look like this

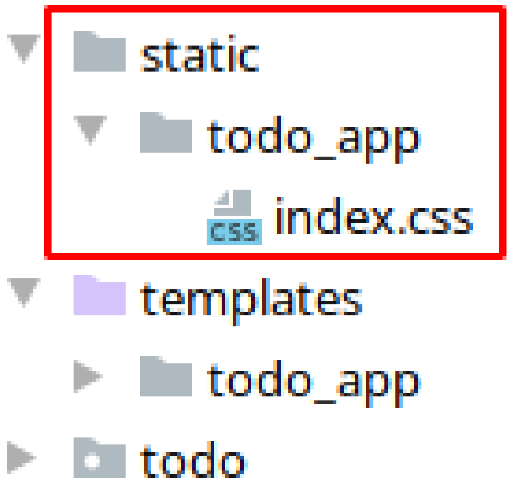


```
VALIDATION SUCCESS
NAME: Peter
AGE: 39
EMAIL: peter@gmail.com
PASSWORD: verysecretpassword
TEXT: Lorem ipsum dolor sit...
```

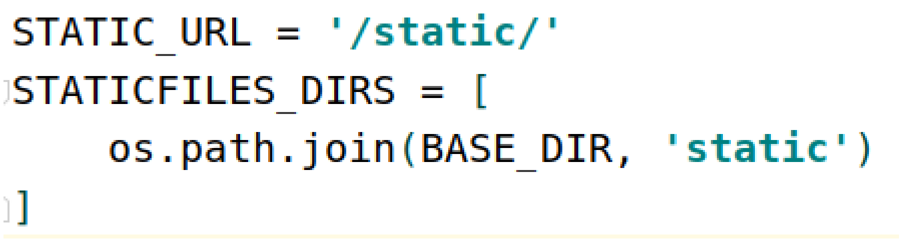
2. Todo App

Using our Todo project from the models lecture, create the forms for our todo. But first, let's do some cleanup.

Move the css into another file directory called **"`static/todo_app/index.css`"**

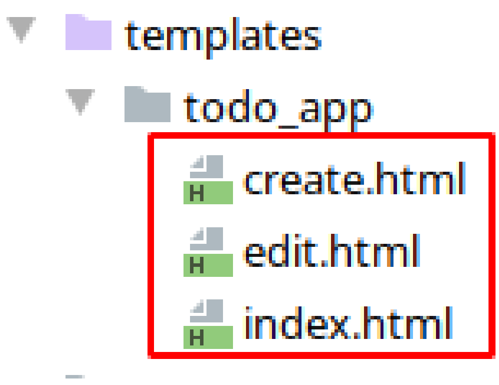


Now we should add that directory to our settings, so we can use it for our application.

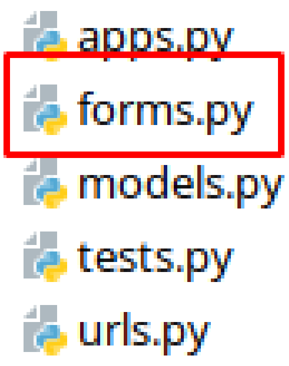


```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]
```

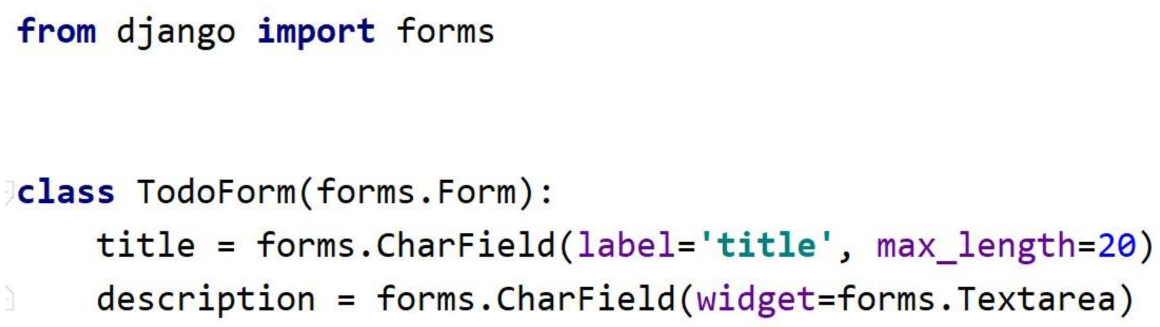
And lastly, add the new html files into the templates folder and get to know what they do and how they work



Now that we're ready to work on our new todo application, let's create our form. Create a file called **"`forms.py`"** in our todo application's folder.



Next, import the forms module from django, as well as our TodoModel and create a form using the fields from the model.



```python
from django import forms


class TodoForm(forms.Form):
    title = forms.CharField(label='title', max_length=20)
    description = forms.CharField(widget=forms.Textarea)
```

And finally, we should implement our views and add the new urls

3. Validation

Using the `forms.py` file from the previous lecture, create a **bot catcher field** and validate each field:

- Name:
  - Must start with an uppercase letter and have a minimum length of 6.
  - You can use a built-in validator for the minimum length.
  - Validation Error message: "`The name must start with an uppercase letter.`"
- Age:
  - Must be bigger than or equal to 0
  - You can use the built-in validator for the validation or create your own
  - Validation Error message: `"The age cannot be less than zero."`
- Email:
  - Contains @ and dot
  - You can use the built-in validator for the validation
  - Validation Error message: `"Enter a valid email."`
- Password:
  - Must have a minimum length of 8 and can only contain letters and numbers.
  - You can use a built-in validator for the minimum length and a custom one for the requirements
  - Validation Error message: `"Enter a valid password."`

You will also need to create a bot catcher field:

- It should not have a requirement.
- It should have a Hidden Input widget.
- The maximum length should be set to zero.
- Message if a bot is caught: **`"This form was created by a bot"`**

```python
def check_for_name(value):
    if not value[0].isupper():
        raise forms.ValidationError("Name needs to start with an uppercase letter")


def check_for_age(value):
    # TODO
    pass


def check_for_password(value):
    # TODO
    pass
```

```python
class FormName(forms.Form):
    name = forms.CharField(validators=[  # TODO
    ])
    age = forms.IntegerField(widget=forms.NumberInput, validators=[  # TODO
    ])
    email = forms.EmailField(widget=forms.EmailInput, validators=[  # TODO
    ])
    password = forms.CharField(widget=forms.PasswordInput, validators=[  # TODO
    ])
    text = forms.CharField(widget=forms.Textarea)
    bot_catcher = forms.CharField(  # TODO
        validators=[  # TODO
        ])

    def clean_bot_catcher(self):
        bot_catcher = ""  # TODO

        if len(bot_catcher) > 0:
            raise forms.ValidationError("GOTCHA BOT!")
        return bot_catcher
```

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

urlpatterns = [
    path('', views.index, name='index'),
    path('create/', views.create, name='create_todo'),
    path('update/<int:todo_id>', views.update, name='update_todo'),
    path('delete/<int:todo_id>', views.delete, name='delete_todo'),
]
```