Validating Forms

Built-in Validators

Form Level Validation

Custom Validator

Field Level Validation

Validating Forms - Form Level Validation

- In forms.py

```
from django import forms
from django.core.exceptions import ValidationError

class ContactForm(forms.Form):

    subject = forms.CharField(max_length=100, required=True)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)


    def myclean(self):
     cleaned_data = super(ContactForm, self).clean()

     email = cleaned_data.get('sender')
     dom = 'gmail.com'

     if dom not in email:
          print('here')
          raise ValidationError('incorrect domain')
```

Validating Forms - Form Level Validation

- In views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from .forms import ContactForm

# Create your views here.
def page(request):

    if request.method == 'POST':

        form = ContactForm(request.POST)
        if form.is_valid():
            form.myclean()
            return HttpResponse('thanks')

    else:

        form = ContactForm()
        context = {'form': form}

        return render(request, 'home.html', context)
```

Validating Forms - Form Level Validation

- In home.html

```
<form method='POST'>
{% csrf_token %}
{{ form }}

<input type="submit" value="submit">
</form>
```

Validating Forms - Field Level Validation

- In forms.py

forms.py

```
from django import forms
from django.core.exceptions import ValidationError

class ContactForm(forms.Form):

    subject = forms.CharField(max_length=100, required=True)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)


    def field_validation(self):
        email = self.cleaned_data.get('sender')
        dom = 'gmail.com'
        if dom not in email:
            raise ValidationError('field validation failed')
```

Validating Forms - Field Level Validation

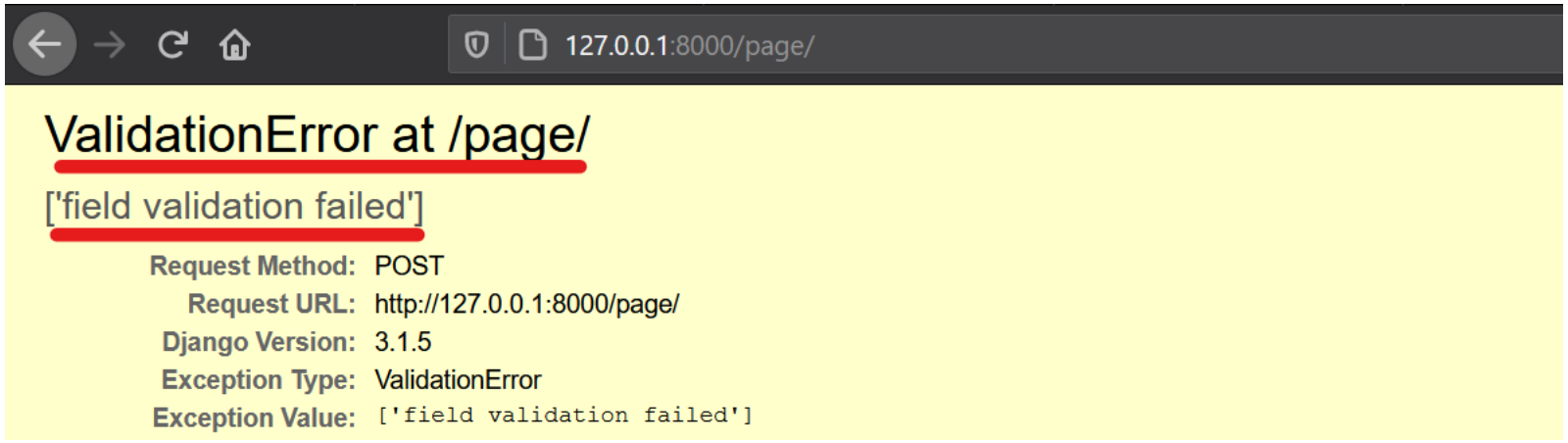
- In forms.py

views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from .forms import ContactForm


def page(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            form.field_validation()
            return HttpResponse('thanks')

    else:
        form = ContactForm()
        context = {'form': form}
        return render(request, 'home.html', context)
```

Validating Forms - Field Level Validation

- If sender has any mail domain other than gmail.com



Forms - Built-in fields

Built-in Fields

- BooleanField
- CharField
- ChoiceField
- TypeChoiceField
- DateField
- DateTimeField
- DecimalField
- DurationField
- EmailField
- FileField
- FilePathField
- FloatField
- ImageField
- IntegerField
- JSONField
- URLField

More:
https://docs.djangoproject.com/en/3.1/ref/forms/fields/#built-in-field-classes

Forms - Built-in fields

forms.py

```
from django import forms

class myForm(forms.Form):
    x1 = forms.BooleanField(required=False)
    x2 = forms.CharField(min_length = 2, max_length = 10,
required=False)

    x = (('s', 'small'),
        ('m', 'medium'))
    x3 = forms.ChoiceField(choices=x)

    x = (('1', '10'),
        ('1', '20'))
    x4 = forms.TypedChoiceField(choices=x,coerce=int)
```

Forms - Built-in fields

forms.py

Contd...

```
    x5 = forms.DateField(required=False)
    x6 = forms.DateTimeField(required=False)
    x7 = forms.DecimalField(required=False)
    x8 = forms.DurationField(required=False)
    x9 = forms.EmailField(required=False)
    x10 = forms.FileField(required=False)
    x11 =
forms.FilePathField(allow_folders=True,path=r'C:\Users\harulp663
\Desktop\practice\myproj', required=False)
    x12 = forms.FloatField(required=False)
    x14 = forms.IntegerField(required=False)
    x15 = forms.JSONField(required=False)
    x16 = forms.URLField(required=False)
```

Forms - Built-in fields

Home.html

```
<form method='POST' enctype="multipart/form-data">
{% csrf_token %}
{{ form }}

<input type="submit" value="submit">
</form>
```

Forms - Built-in fields

Views.py

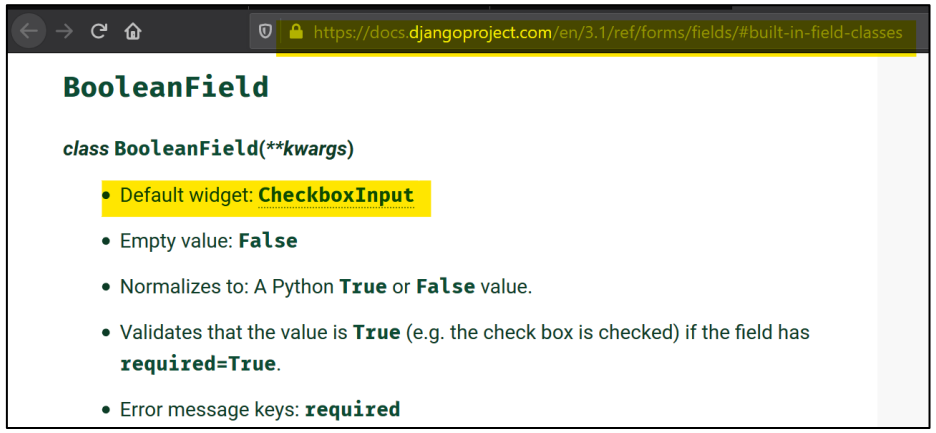
```
from django.shortcuts import render
from django.http import HttpResponse

from .forms import myForm


def page(request):
     if request.method == 'POST':
          form = myForm(request.POST,request.FILES)
          if form.is_valid():
               values = form.clean()
               print(values)
               return HttpResponse('check ur command prompt')
          else:
               return HttpResponse('not valid form')

     else:
          form = myForm()
          context = {'form': form.as_ul}
          return render(request, 'home.html', context)
```

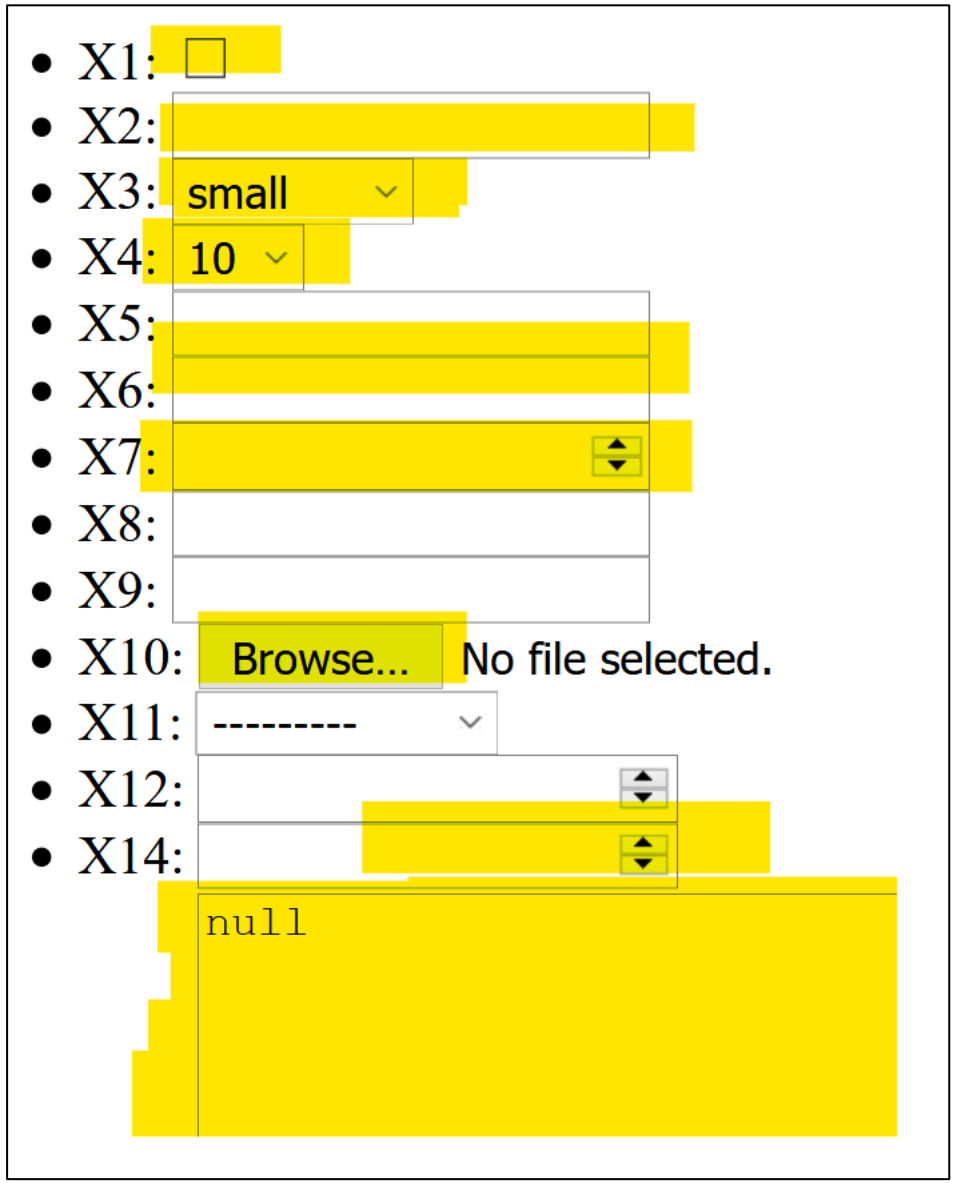
Forms - Built-in widgets

Built-in widgets

Highlighted some widgets in the diagram

- Whenever a built-in field is used, there is a default widget used





Forms - Built-in widgets

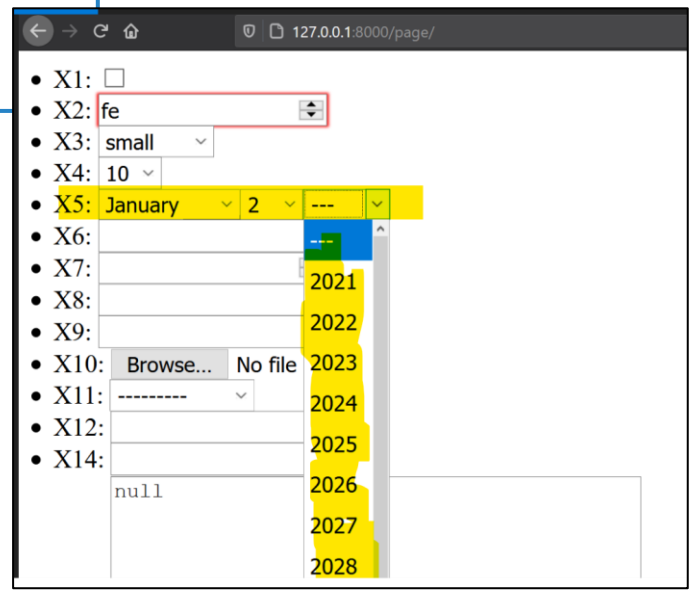
- **Widgets handling input of text**
  - TextInput
  - NumberInput
  - EmailInput
  - URLInput
  - PasswordInput
  - HiddenInput
  - DateInput
  - DateTimeInput
  - TimeInput
  - TextArea
- **Selector and checkbox widgets**
  - CheckBoxInput
  - Select
  - NullBooleanSelect
  - SelectMultiple
  - RadioSelect
  - CheckboxSelectMultiple
- **File upload widgets**
  - File Input
  - ClearableFileInput

More:
https://docs.djangoproject.com/en/3.1/ref/forms/widgets/#built-in-widgets

Forms - Built-in widgets

forms.py

```
from django import forms

class myForm(forms.Form):
    x5 = forms.DateField( required=False,
                          widget=forms.SelectDateWidget)
```



Model form

- For database-driven app
- Using models to create a form

models.py

```
from django.db import models

# Create your models here.
class member(models.Model):
    first_name= models.CharField(max_length=30)
    last_name= models.CharField(max_length=30)
```

Model form

forms.py

```
from django.forms import ModelForm
from .models import member


class myModelForm(ModelForm):
    class Meta:
        model = member
        fields = ["first_name", "last_name"]
```

Model form

views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from .forms import myModelForm
from .models import member


def page(request):
    if request.method == 'POST':
        form = myModelForm(request.POST)
        if form.is_valid():
            form.save()
            values = form.clean()
            print(values)
            return HttpResponse('check ur command prompt')
        else:
            return HttpResponse('not valid form')

    else:
        form = myModelForm()
        context = {'form': form}
        return render(request, 'home.html', context)
```

Model form

- You can save the form data to the DB

views.py

```
form = myModelForm(request.POST)
if form.is_valid():
    form.save()
```

- Suppose update an existing record in db?

views.py

```
x = member.objects.get(first_name="BEULAH S")
form = myModelForm(request.POST, instance=x)
if form.is_valid():
    form.save()
```

Form sets

- Suppose you need the same form to be displayed n times – use formset

views.py

```
from django.forms import formset_factory

myFormSet = formset_factory(myModelForm, extra=2)
context = {'form': myFormSet}
return render(request, 'home.html', context)
```

- Output