AceleraDev Loadsmart Women Edition

Módulo 4

Camila Maia

Esta apresentação está disponível em:

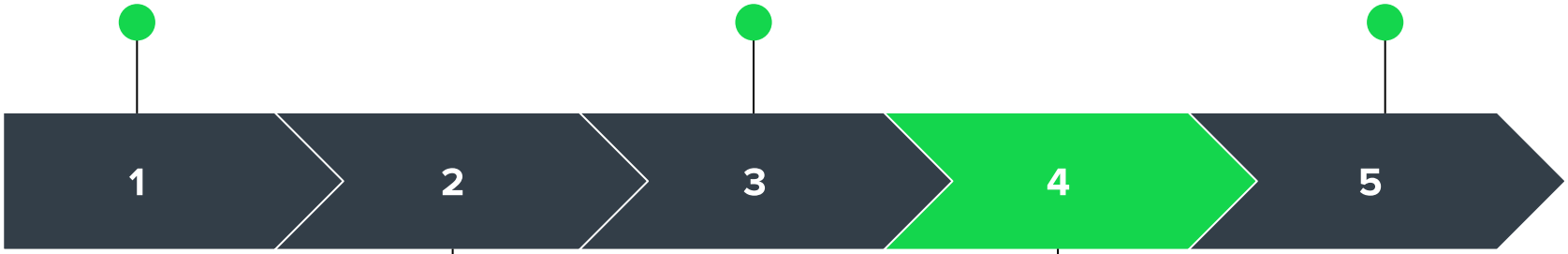
https://github.com/camilamaia/acelera-dev-loadsmart-women/

Módulos

Python: Noções básicas
Primeiro programa, teste, estrutura de dados, condicionais, repetições, operações, funções, classes, objetos...

Django I: instalação, iniciando um projeto, arquitetura, modelos, views, django admin, URLS

REST APIs: O protocolo HTTP, Rest APIs e Django REST Framework



Um pouco mais de Python
Exceções, decorators, list e dict comprehensions, map, reduce, filter...

Django II: Templates, HTML e CSS básico, Forms, Autorização e Autenticação.

HTML

- Hyper Text Markup Language
- HTML é a linguagem usada para criação de sites. Tem como função exibir as informações.
- A base do HTML são suas **tags** e **atributos**.

HTML Tags

- Conjunto de caracteres que formam elementos
- São a representação de um elemento html
- Browsers do not display the HTML tags, but use them to render the content of the page

Exemplos:

- <h1>, <p>, <img>, <a>

Most tags must be opened <h1> and closed </h1> in order to function.

- The `<!DOCTYPE html>` declaration defines this document to be HTML5
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the document
- The `<title>` element specifies a title for the document
- The `<body>` element contains the visible page content
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

```
<html>

  <head>

    <title>Page title</title>

  </head>

  <body>

    <h1>This is a heading</h1>

    <p>This is a paragraph.</p>

    <p>This is another paragraph.</p>

  </body>

</html>
```

HTML Atributos

- Informações passadas dentro da Tag para que ela se comporte da maneira esperada

Exemplos:

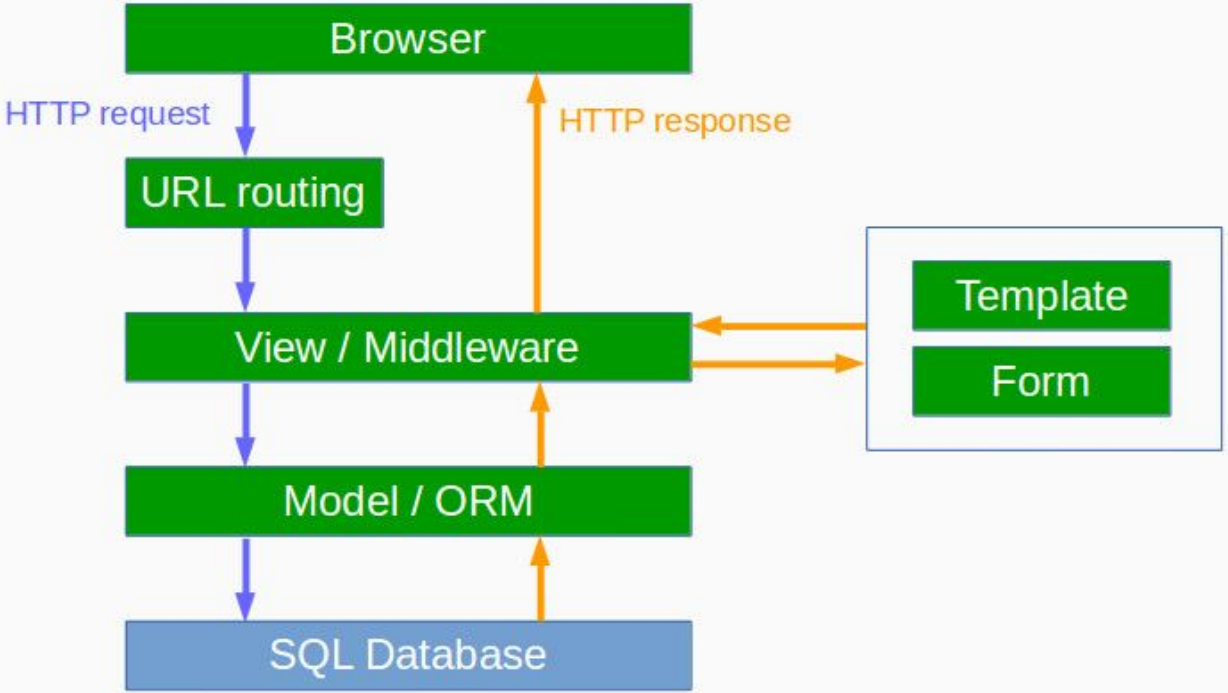
- class, id, href, src

CSS

- CSS é a linguagem usada para **estilizar** uma página HTML.
- No CSS são definidas cores, fontes, margens, etc.


django

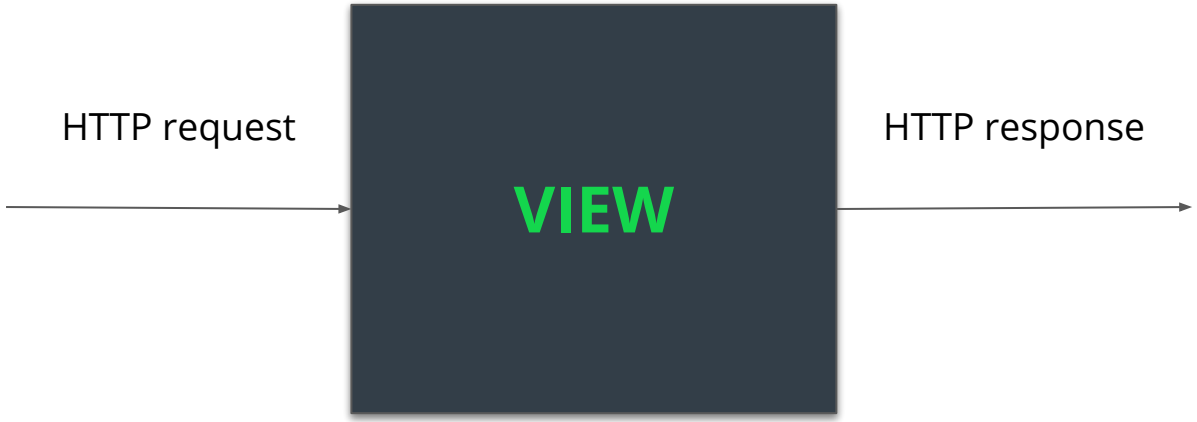
Django's architecture



View

- Descreve **qual** dado será apresentado
- A view is simply a Python function that takes a Web request and returns a Web response
- This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image . . . or anything, really
- The view itself contains whatever arbitrary logic is necessary to return that response

View



Function Based Views

```
url(r'^(?P<question_id>[0-9]+)/vote/$',
    views.vote,
    name='vote'
),
```

```
def vote(request, question_id):
    if request.method == 'GET':
        pass
    if request.method == 'POST':
        pass
    if request.method == 'PUT':
        pass
    if request.method == 'DELETE':
        pass
```

Sempre retornando uma HTTP Response

Function Based Views

- Nomes para as urls são importantes!
  - Refactoring
  - Evitar urls quebradas

Class Based Views

- Class-based views provide an alternative way to implement views as Python **objects** instead of functions.
- They do not replace function-based views
- Reutilizar código através de:
  - Heranças
  - Mixins
- Não precisar fazer um branch de if's

Class Based Views

```python
# urls.py
from django.urls import path
from myapp.views import MyView

urlpatterns = [
    path('about/', MyView.as_view()),
]
```

```python
from django.http import HttpResponse
from django.views import View

class GreetingView(View):
    greeting = "Good Day"

    def get(self, request):
        return HttpResponse(self.greeting)
```

Sempre retornando uma HTTP Response

Class Based Generic Views

- Django já provê uma série de classes e mixins para serem utilizados
- ListView
- DetailView
- UpdateView
- CreateView
- DeleteView

```
class ResultsView(generic.DetailView):
    model = Question
    template_name = 'polls/results.html'
```

FBVs ou CBVs?

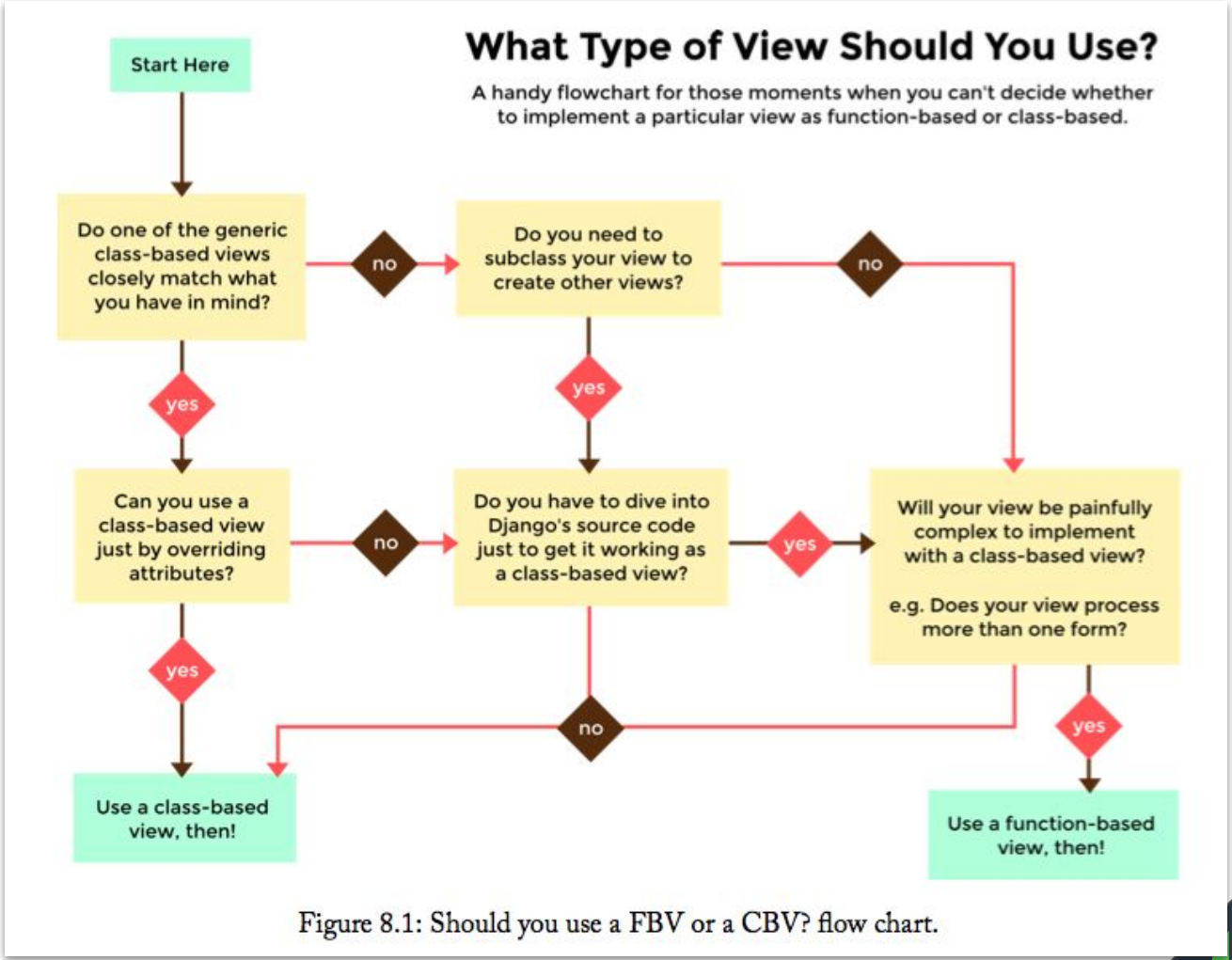
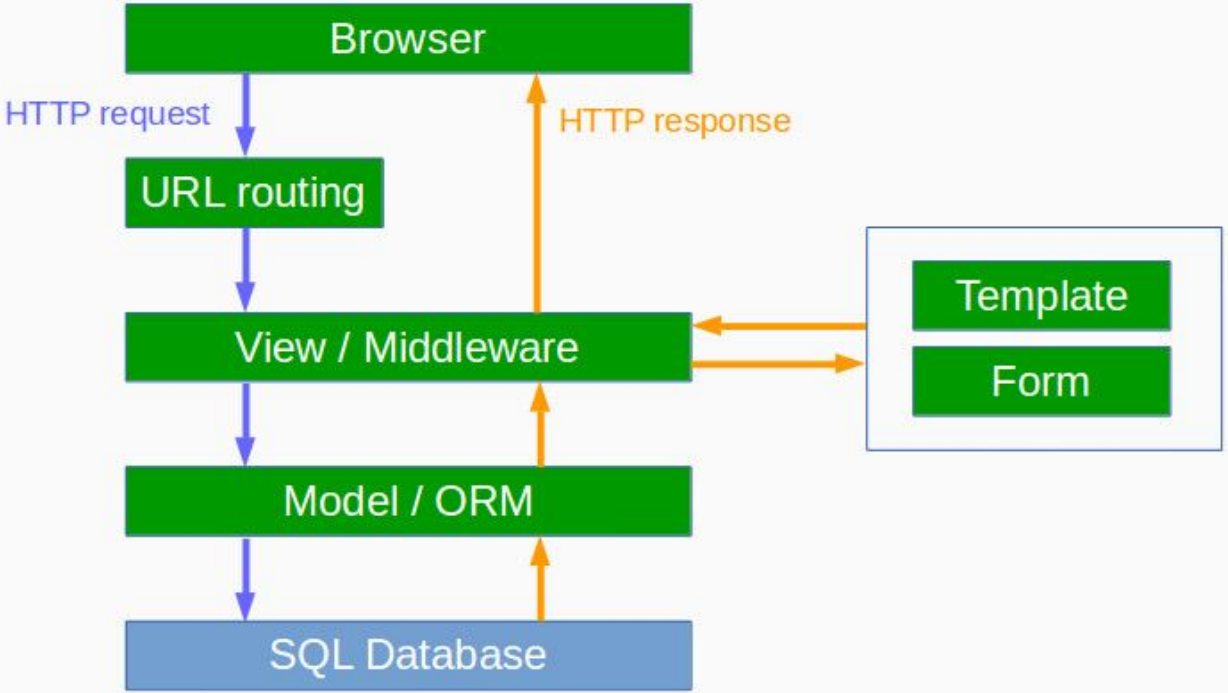


Figure 8.1: Should you use a FBV or a CBV? flow chart.

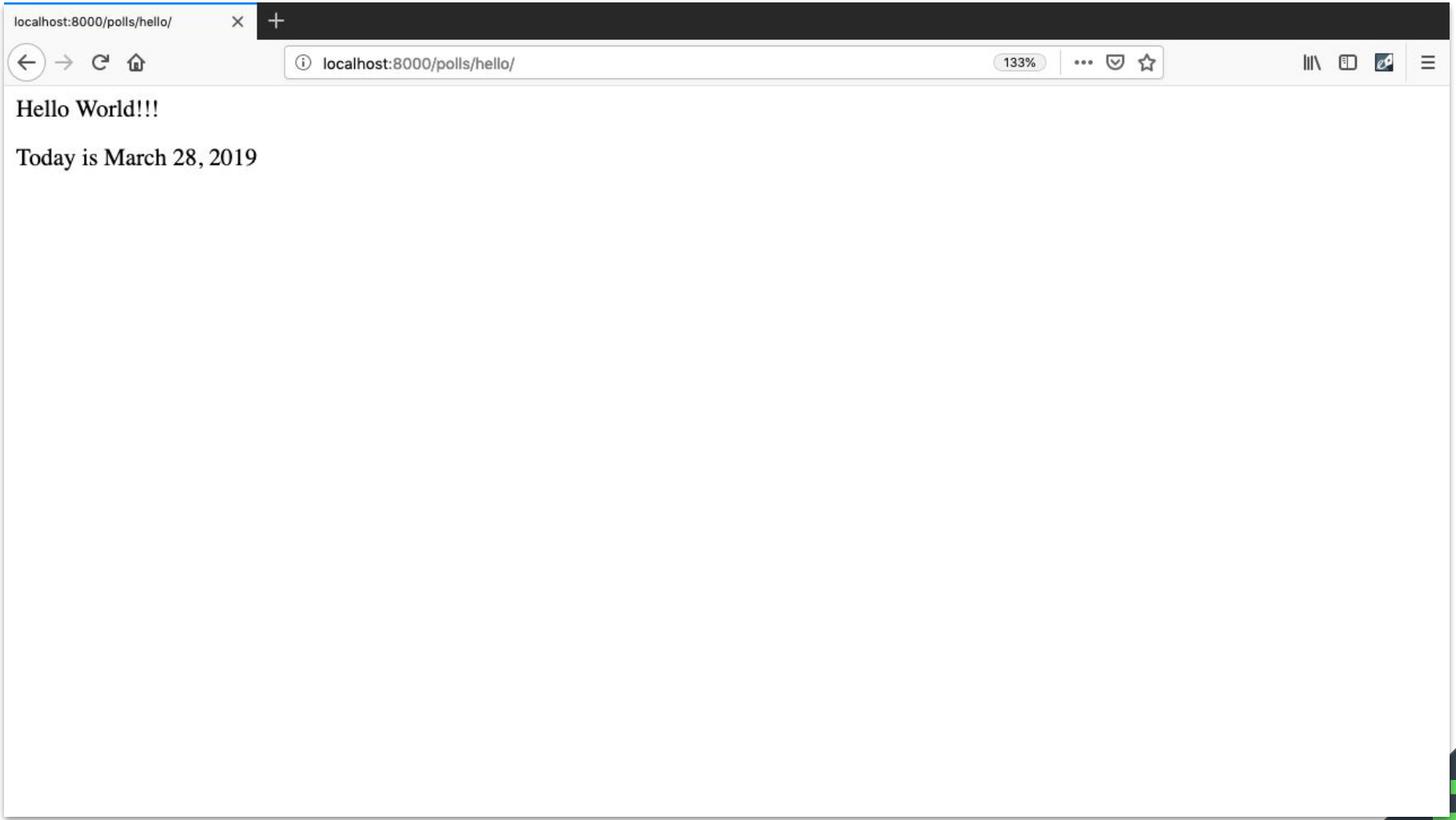
https://www.twoscoopspress.com/products/two-scoops-of-django-1-11

Django's architecture



Django Templates

- Django is a web framework
- Django helps to generate HTML dynamically


localhost:8000/polls/hello/
localhost:8000/polls/hello/
133%
Hello World!!!
Today is March 28, 2019

Django Templates

static parts of the desired HTML output

+

some special syntax describing how **dynamic** content will be inserted.

Django Templates

Django Template Language (DTL)

- Variables
  - {{variable}}
- Filters
- Tags
  - if
  - for
  - block / extends
  - comment

Django Templates

- Render Function
  - Request − The initial request.
  - The path to the template − This is the path relative to the TEMPLATE_DIRS option in the project settings.py variables.
  - Dictionary of parameters − A dictionary that contains all variables needed in the template. This variable can be created or you can use locals() to pass all local variable declared in the view.

Django Templates

```python
def hello(request):
    if request.method == 'GET':
        today = datetime.datetime.now().date()
        return render(
            request,
            "polls/hello.html",
            {"today": today}
        )
```

```python
class ResultsView(generic.DetailView):
    model = Question
    template_name = 'polls/results.html'
```

Django Forms

- Django provides a range of tools and libraries to help you build forms to **accept input** from site visitors, and then **process** and **respond to the input**.

HTML Forms

- A form is a collection of elements inside **<form>...</form>**
- Forms allow a visitor to do things like:
  - Enter text,
  - Select options,
  - Manipulate objects or controls,
  - ...
- Then send that information back to the **server**.

HTML Forms

- Action
  - The action attribute defines the action to be performed when the form is submitted.
- Input
  - The <input> element can be displayed in several ways, depending on the type attribute.

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_radio

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit

HTML Forms

- Method
  - The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_get

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_post

Como queremos

```
<form action="/your-name/" method="post">
    <label for="your_name">Your name: </label>
    <input id="your_name" type="text" name="your_name" value="
{{ current_name }}">
    <input type="submit" value="OK">
</form>
```

Forms

forms.py

```
from django import forms

class NameForm(forms.Form):
    your_name = forms.CharField(label='Your name',
max_length=100)
```

Django Forms

- A Form instance has an **is_valid()** method, which runs validation routines for all its fields.
- When this method is called, if all fields contain valid data, it will:
  - return True
  - place the form's data in its **cleaned_data** attribute.

Rendered

```
<label for="your_name">Your name: </label>
<input id="your_name" type="text" name="your_name"
maxlength="100" required>
```

Note that it does not include the <form> tags, or a submit button. We'll have to provide those ourselves in the template.

View

views.py

```python
from django.http import HttpResponseRedirect
from django.shortcuts import render

from .forms import NameForm

def get_name(request):
    # if this is a POST request we need to process the form data
    if request.method == 'POST':
        # create a form instance and populate it with data from the
request:
        form = NameForm(request.POST)
        # check whether it's valid:
        if form.is_valid():
            # process the data in form.cleaned_data as required
            # ...
            # redirect to a new URL:
            return HttpResponseRedirect('/thanks/')

    # if a GET (or any other method) we'll create a blank form
    else:
        form = NameForm()

    return render(request, 'name.html', {'form': form})
```

Template

```
<form action="/your-name/" method="post">
    {% csrf_token %}
    {{ form }}
    <input type="submit" value="Submit">
</form>
```

Models e Forms

- Null -- Banco de Dados
  - Default is False
- Blank -- Forms
  - Default is False

```
class Question(models.Model):
    question_text = models.CharField(
        max_length=200,
        null=True,
        blank=True
    )
```

null=True
blank=True

Field Type	Setting null=True	Setting blank=True
`CharField`, `TextField`, `SlugField`, `EmailField`, `CommaSeparatedInteger-Field`, `UUIDField`	*Okay* if you also have set both `unique=True` and `blank=True`. In this situation, `null=True` is required to avoid unique constraint violations when saving multiple objects with blank values.	*Okay* if you want the corresponding form widget to accept empty values. If you set this, empty values are stored as `NULL` in the database if `null=True` and `unique=True` are also set. Otherwise, they get stored as empty strings.
`FileField`, `ImageField`	*Don't do this.* Django stores the path from `MEDIA_ROOT` to the file or to the image in a CharField, so the same pattern applies to FileFields.	*Okay.* The same pattern for CharField applies here.

null=True
blank=True

Field Type	Setting null=True	Setting blank=True
`BooleanField`	*Don't do this.* Use `NullBooleanField` instead.	*Don't do this.*
`IntegerField`, `FloatField`, `DecimalField`, `DurationField`, etc	*Okay* if you want to be able to set the value to `NULL` in the database.	*Okay* if you want the corresponding form widget to accept empty values. If so, you will also want to set `null=True`.
`DateTimeField`, `DateField`, `TimeField`, etc.	*Okay* if you want to be able to set the value to `NULL` in the database.	*Okay* if you want the corresponding form widget to accept empty values, or if you are using auto_now or auto_now_add. If it's the former, you will also want to set `null=True`.
`ForeignKey`, `ManyToManyField`, `OneToOneField`	*Okay* if you want to be able to set the value to `NULL` in the database.	*Okay* if you want the corresponding form widget (e.g. the select box) to accept empty values. If so, you will also want to set `null=True`.
`GenericIPAddressField`	*Okay* if you want to be able to set the value to `NULL` in the database.	*Okay* if you want to make the corresponding field widget accept empty values. If so, you will also want to set `null=True`.

Dúvidas?

Conhecimento Passado

Legal, mas não aprendi nada novo, não.

Aprendi muita coisa nova!

Conhecimento: 0-10

Velocidade

5: Velocidade ideal.

ZzzZzzz, pode acelerar isso aí.

Muito rápido, tô assimilando o primeiro slide ainda.

Velocidade: 0-10

Conteúdos

- https://www.tutorialspoint.com/django/django_template_system.htm
- https://docs.djangoproject.com/en/2.1/topics/forms/
- https://docs.djangoproject.com/en/2.1/topics/templates/
- https://www.twoscoopspress.com/products/two-scoops-of-django-1-11
- https://docs.djangoproject.com/en/2.1/topics/class-based-views/generic-display/
- https://docs.djangoproject.com/en/2.1/topics/class-based-views/
- https://easyaspython.com/mixins-for-fun-and-profit-cb9962760556
- https://djangobook.com/user-authentication-django/
- https://www.w3schools.com/html/html_intro.asp
-

MUITO OBRIGADA!