

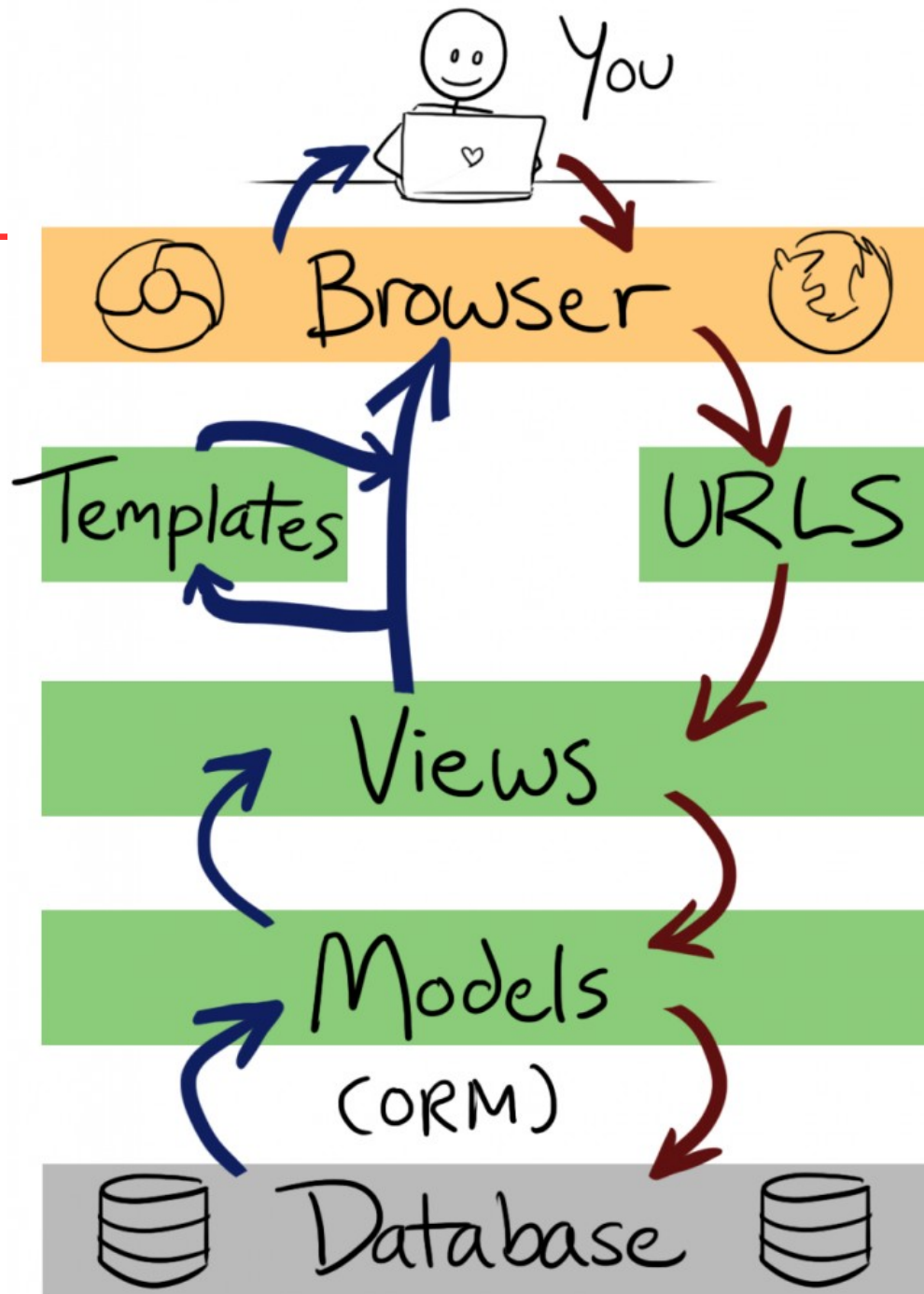
Tópicos Especiais em Sistemas para Internet III

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Agenda

- Django
 - O fluxo MTV
 - Django Admin
- O Projeto

Fluxo MTV



Ativando o Ambiente Virtual

Vamos começar ... No Windows

No prompt de comando (cmd), digite:

- pip install virtualenv
- virtualenv venv
- cd venv
- cd Scripts
- Activate.bat
- (venv)

Vamos começar ...

No Linux

No terminal, digite:

- `virtualenv -p python3 venv`
- `source venv/bin/activate`
- `(venv)`

Arquivo requirements.txt

```
pip install -r requirements.txt
```

Projeto Feriado v. 0.6

Uma das partes mais poderosas do Django é a interface de administração automática.

Ele lê os metadados de seus modelos para fornecer uma interface rápida e centrada no modelo, na qual usuários confiáveis podem gerenciar o conteúdo do seu site.

O uso recomendado pelo administrador é limitado à ferramenta de gerenciamento interno de uma organização. **Não se destina a criar todo o seu front end.**

Fonte: <https://docs.djangoproject.com/pt-br/3.0/ref/contrib/admin/>

Django Admin



127.0.0.1:8000/admin/login/?next=/admin/



Django administration

Username:

Password:

Log in

`python manage createsuperuser`

- Digite usuário
- Digite senha
 - O Django possui validador de senha

Django Admin



→ ↻ ⓘ 127.0.0.1:8000/admin/ 🔍 ☆

Django administration WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PA](#)

Site administration

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add ✎ Change
Users	+ Add ✎ Change

Recent actions

My actions

None available

admin.py (do app)

```
from django.contrib import admin
from .models import FeriadoModel

admin.site.register(FeriadoModel)
```

Django Admin

→ ↻ ⓘ 127.0.0.1:8000/admin/

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	✎ Change
Users	+ Add	✎ Change

FERIADO

Feriado models	+ Add	✎ Change
-----------------------	-----------------------	--------------------------

apps.py (do app)

```
from django.apps import AppConfig

class FeriadoConfig(AppConfig):
    name = 'feriado'
    verbose_name = 'Controle de Feriados'
```

setting.py (do projeto)

...

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

...

Django Admin

→ ↻ ⓘ 127.0.0.1:8000/admin/

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

[+ Add](#) [✎ Change](#)

Users

[+ Add](#) [✎ Change](#)

CONTROLE DE FERIADOS

Feriado models

[+ Add](#) [✎ Change](#)

Django Admin

→ ↻ ⓘ 127.0.0.1:8000/admin/

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

[+ Add](#)

[✎ Change](#)

Users

[+ Add](#)

[✎ Change](#)

CONTROLE DE FERIADOS

Feriado models

[+ Add](#)

[✎ Change](#)



```
class FeriadoModel(models.Model):
    nome = models.CharField('Feriado', max_length=50)
    dia = models.IntegerField('Data')
    mes = models.IntegerField('Mês')
    modificado_em = models.DateTimeField(
        verbose_name='modificado em',
        auto_now_add=False, auto_now=True)

    def __str__(self):
        return self.nome

class Meta:
    verbose_name_plural = 'Feriados'
    verbose_name = 'Feriado'
```

Customizando o Django Admin

O Admin do Django é genérico, mas customizável.

Para isso, vamos criar um classe que vai gerir a comunicação do modelo *FeriadoModel* com o Django Admin.

- Criar a classe `FeriadoModelAdmin` no arquivo `admin.py`.

```
from django.contrib import admin
from .models import FeriadoModel
```

```
class FeriadoModelAdmin(admin.ModelAdmin):
    list_display = ('nome', 'dia', 'mes', 'modificado_em')
    date_hierarchy = 'modificado_em'
    search_fields = ('nome', 'dia', 'mes')
```

```
admin.site.register(FeriadoModel, FeriadoModelAdmin)
```

Customizando o Django Admin

Criar o método `registrado_esse_ano` para auxiliar exibir apenas os feriados criados esse ano.

```
from datetime import date
```

```
...
```

```
class FeriadoModelAdmin(admin.ModelAdmin):  
    list_display = ('nome', 'dia', 'mes', 'modificado_em',  
        'registrado_esse_ano')  
    date_hierarchy = 'modificado_em'  
    search_fields = ('nome', 'dia', 'mes')
```

```
def registrado_esse_ano(self, obj):  
    hoje = date.today()  
    return obj.modificado_em.year == hoje.year
```

```
class FeriadoModelAdmin(admin.ModelAdmin):  
    list_display = ('nome', 'dia', 'mes', 'modificado_em',  
                    'registrado_esse_ano')  
    date_hierarchy = 'modificado_em'  
    search_fields = ('nome', 'dia', 'mês')  
    list_filter = ('modificado_em',)  
  
    def registrado_esse_ano(self, obj):  
        hoje = date.today()  
        return obj.modificado_em.year == hoje.year  
  
    registrado_esse_ano.short_description = 'Registrado este ano'  
    registrado_esse_ano.boolean = True
```


Customizando o Django Admin



Exibir os dados em ordem crescente / decrescente.

```
class FeriadoModel(models.Model):  
    ...  
  
    class Meta:  
        verbose_name_plural = 'Feriados'  
        verbose_name = 'Feriado'  
        ordering = ('mes', '-dia')
```

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Criar um novo app no projeto e visitar os arquivos de um app Django.

Como criar um teste para checar se a ordenação está funcionando corretamente ?

Conhecer o projeto **Django-Jazzmin**

<https://github.com/farridav/django-jazzmin>