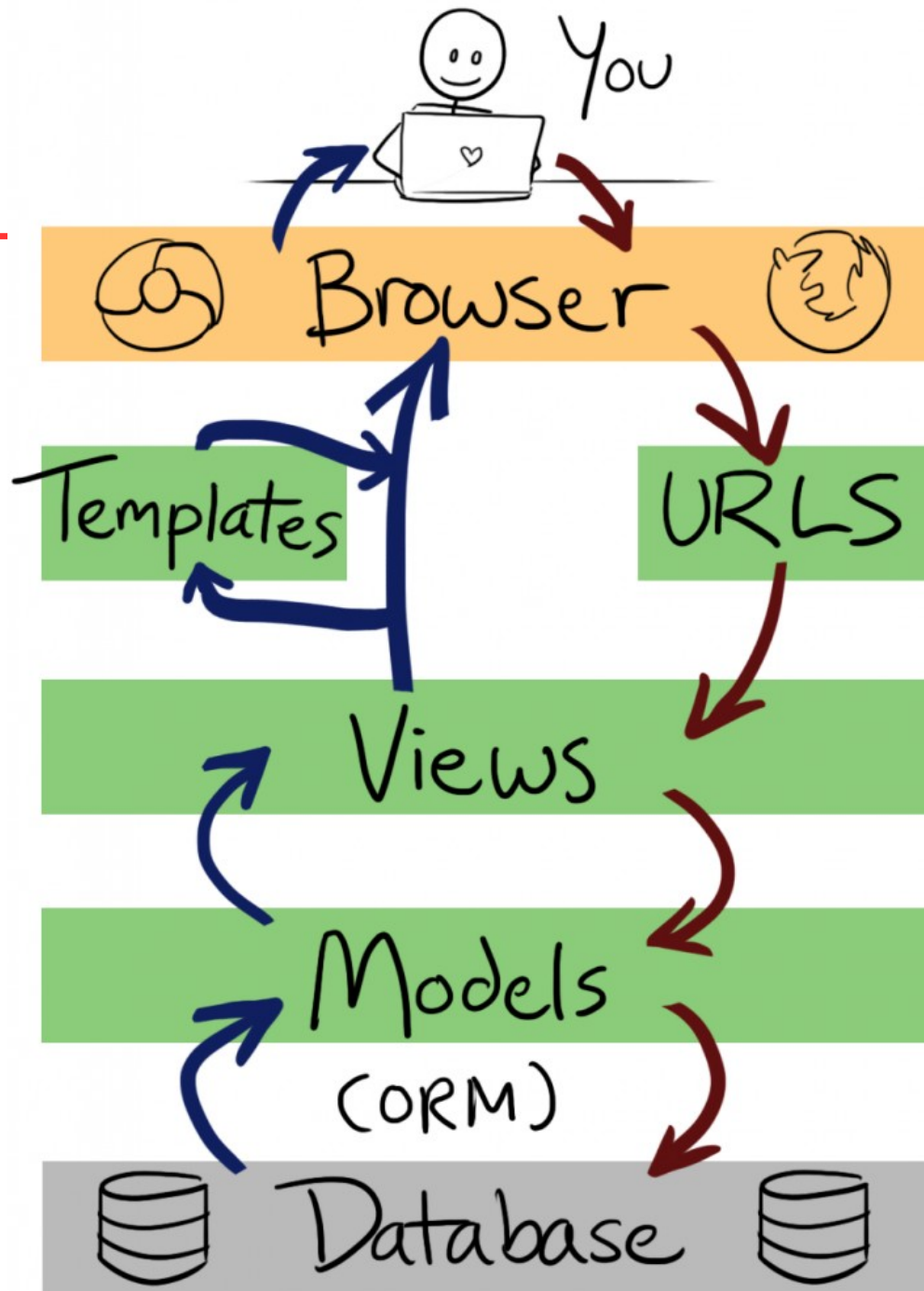


Tópicos Especiais em Sistemas para Internet III

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

- Django
 - O fluxo MTV
 - Atualização do primeiro projeto, com testes
 - Modelos
- 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 django==2.2.10
```

```
pip install ipython
```

```
pip install ipdb
```

```
pip freeze >> requirements.txt
```

Projeto Feriado v. 0.4

Um modelo é a fonte única e definitiva de informações sobre seus dados.

- Cada modelo é uma classe Python, uma subclasse de **django.db.models.Model**.
- Cada atributo representa um campo do banco de dados.

Fonte: <https://docs.djangoproject.com/en/3.0/topics/db/models/>

```
from django.db import models
```

```
class FeriadoModel(models.Model):  
    nome = models.CharField('Feriado',max_length=50)  
    dia = models.IntegerField('Data')  
    mes = models.IntegerField('Mês')  
  
    def __str__(self):  
        return self.nome
```

Executar a migração

Execute:

- `python manage.py makemigrations`
- `python manage.py migrate`

O que aconteceu ?

Editar models.py (do app)

```
from django.db import models

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
```

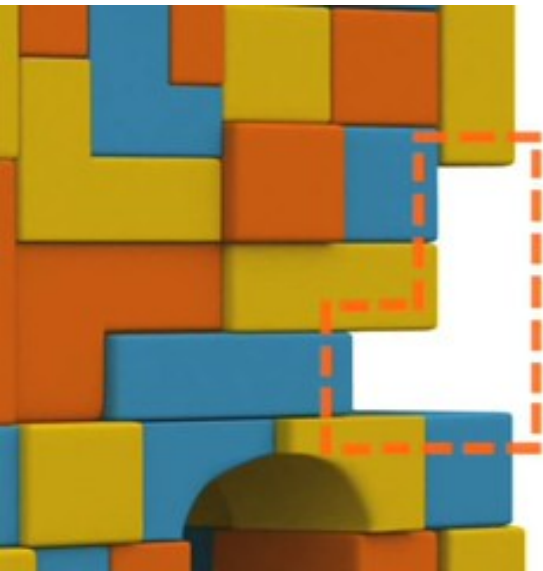
Executar a migração

Execute:

- `python manage.py makemigrations`
- `python manage.py migrate`

O que aconteceu ?

Test it !



Teste unitário

/feriado/tests.py (continuação)

```
from feriado.models import FeriadoModel  
from datetime import datetime
```

```
class FeriadoModelTest(TestCase):  
    def setUp(self):  
        self.feriado = 'Natal'  
        self.mes = 12  
        self.dia = 25  
        self.cadastro = FeriadoModel(  
            nome=self.feriado,  
            dia=self.dia,  
            mes=self.mes,  
        )  
        self.cadastro.save()
```

Teste unitário

/feriado/tests.py (continuação)

```
def test_created(self):
    self.assertTrue(FeriadoModel.objects.exists())

def test_modificado_em(self):
    self.assertIsInstance(self.cadastro.modificado_em, datetime)

def test_nome_feriado(self):
    nome = self.cadastro.__dict__.get('nome', '')
    self.assertEqual(nome, self.feriado)

def test_dia_feriado(self):
    dia = self.cadastro.__dict__.get('dia', '')
    self.assertEqual(dia, self.dia)
```


ORM

Object-Relational Mapping

Object Relational Mapping, ou relacionamento objeto-relacional é a técnica que mapeia o uso do banco de dados, convertendo-os em objetos.

Brincando com o ORM

O método save

Execute:

- `python manage.py shell`
 - `from feriado.models import FeriadoModel`
 - `f = FeriadoModel(nome="Natal",dia=25,mes=12)`
 - `f.save()`
 - `f.pk`

O que aconteceu ?

Brincando com o ORM

```
In [1]: from feriado.models import FeriadoModel
```

```
In [2]: f = FeriadoModel(nome="Natal", dia=25, mes=12)
```

```
In [3]: f
```

```
Out[3]: <FeriadoModel: Natal>
```

```
In [4]: f.id
```

```
In [5]: f.save()
```

Brincando com o ORM

O método save

```
from feriado.models import FeriadoModel  
  
f.nome = "Páscoa"  
  
f.save()  
  
f2 = FeriadoModel.objects.get(pk=1)  
  
f2.id = None  
  
f2.save()
```

O método **save()** realiza update quando identifica id com algum valor, e realiza um create quando id estiver nulo.

Brincando com o ORM

O método create



```
FeriadoModel.objects.all()
f.delete()
FeriadoModel.objects.all()
f3= FeriadoModel.objects.create(
    nome='Tiradentes',dia=21,mes=4
)
```

O método **create()** salva o registro no banco e retorna o objeto.

Brincando com o ORM

O método get



```
FeriadoModel.objects.all()
```

```
f3 = FeriadoModel.objects.get(pk=2)
```

```
f4 = FeriadoModel.objects.get(nome__contains='a')
```

O método **get()** retorna apenas uma instância. A segunda forma ocorre erro: `MultipleObjectsReturned`.

`type(f3) => Instância do modelo.`

`type(FeriadoModel) => Descreve a tabela`

`type(FeriadoModel.objects) => Manager manipula a tabela`

`type(FeriadoModel.objects.all()) => QuerySet`

Brincando com o ORM

QuerySet

```
qs = FeriadoModel.objects.all()
```

```
for feriado in qs:  
    print(feriado)
```

```
qs = FeriadoModel.objects.all()  
qs = qs.filter(nome__contains='a')  
qs = qs.filter(nome__contains='P')  
qs
```

Brincando com o ORM

QuerySet - continuação



```
qs = FeriadoModel.objects.all().update(dia=15)
```

```
qs = FeriadoModel.objects.all()
```

```
qs.delete()
```

Brincando com o ORM

Outra forma de acessar os dados



```
from django.db import connection
```

```
cursor = connection.cursor()
```

```
cursor.execute('select * from feriado_feriadomodel')
```

```
cursor.fetchone()
```

```
cursor.execute('select * from feriado_feriadomodel WHERE  
nome =%s', ['Tiradentes']) <== NÃO FAÇA CONCATENAÇÃO DE STRING
```

```
cursor.fetchone()
```

Comandos do manage.py

Comandos do manage.py

python manage.py flush

Apagar todos os dados do banco de dados

python manage.py inspectdb

Gerar classes baseado em uma inspeção no banco de dados.

python manage.py makemigrations

Cria um arquivo migrate, arquivos de migração do esquema do banco de dados.

python manage.py migrate

Executa a migração no banco de dados.

Dúvidas

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

Ajustar o projeto, para que exista apenas uma view.

- Esta deve buscar no banco de dados se hoje é feriado ou não, baseado nos feriados cadastrados no banco de dados.
- Se hoje for feriado, apresente qual feriado é hoje.