



Live de Python #160



1. O peewee

Uma introdução

2. O ORM

Entendendo um pouco sobre orms

3. Queries

Como achar os dados?

4. Playhouse

Extensões do peewee



picpay.me/dunossauro



apoia.se/livedepython



PIX



Ajude o projeto



Ademar Peixoto, Alex Lima, Alexandre Harano, Alexandre Santos, Alexandre Tsuno, Alysson Oliveira, Amaziles Carvalho, Andre Rodrigues, André Rocha, Arnaldo Turque, Bruno Oliveira, Caio Nascimento, César Almeida, César Moreira, Davi Ramos, David Kwast, Diego Guimarães, Dilenon Delfino, Douglas Bastos, Edgard Sampaio, Elias Soares, Érico Andrei, Eugenio Mazzini, Everton Alves, Fabio Barros, Fabio Castro, Fabrício Coelho, Flavkaze, Franklin Silva, Fábio Serrão, Gabriel Simonetto, Gabriel Soares, Gabriela Santiago, Geandreson Costa, Guilherme Felitti, Guilherme Marson, Guilherme Ostrock, Gustavo Chacon, Henrique Machado, Italo Silva, Johnny Tardin, Jonatas Leon, Jonatas Oliveira, Jorge Plautz, José Prado, João Lugão, João Schiavon, Juan Gutierrez, Jônatas Silva, Júlia Kastrup, Kaneson Alves, Leonardo Cruz, Leonardo Galani, Leonardo Mello, Lidianne Monteiro, Lorena Ribeiro, Lucas Barros, Lucas Ferreira, Lucas Mello, Lucas Mendes, Lucas Teixeira, Lucas Valino, Luiz Lima, Maiquel Leonel, Maiquel Leonel, Marcela Campos, Marcelo Rodrigues, Maria Clara, Melissa Mendonça, Moisés Andrade, Natan Cervinski, Nicolas Teodosio, Patric Lacouth, Patricia Minamizawa, Patrick Gomes, Paulo Tadei, Pedro Andrade, Pedro Pereira, Peterson Santos, Rafael Lino, Reinaldo Silva, Rodrigo Ferreira, Rodrigo Vaccari, Ronaldo Silva, Rubens Gianfaldoni, Sandro Mio, Silvio Xm, Thiago Araujo, Thiago Borges, Thiago Bueno, Tyrone Damasceno, Valdir Junior, Victor Geraldo, Vinícius Bastos, Vinícius Ferreira, Vítor Gomes, Wendel Rios, Wesley Mendes, Willian Lopes, Willian Lopes, Willian Rosa, Wilson Duarte



Obrigado você



Uma introdução

pee
wee

Peewee



Peewee é um ORM simples e pequeno (1 arquivo). Possui poucos (mas expressivos) conceitos. O que o torna fácil de aprender e intuitivo de usar

- licença MIT
- Versão atual: 3.14.3
- 11 anos de história (22/12/2010)



Quais bancos o Peewee suporta?

- Sqlite
- PostgreSQL
- mySQL
- CockroachDB
- * Datatable *

Qual a cara de um código com Peewee?



```
1  from peewee import SqliteDatabase, Model, CharField, DateField
2
3  db = SqliteDatabase('pessoas.db')
4
5  class Pessoa(Model):
6      nome = CharField()
7      aniversário = DateField()
8
9      class Meta:
10         database = db  # linha 3
```


`pip install peewee`



Bora instalar?



Pq não SQLAlchemy?



[deleted] 4 years ago · edited 4 years ago

I'm the author of Peewee so as you might expect, I'm much more familiar with it's strengths and weaknesses. I can say, however, that SQLAlchemy is the gold standard for ORM in the Python world. It has a very active community and a maintainer who is committed to excellence. If you're a glass-half-empty guy, to put it another way, you can't go wrong if you choose SQLAlchemy.

Why would you choose Peewee, then?

- It's damn small, and easy to pick up and use. I think this is the #1 reason people use it.
- I've worked very hard to provide consistent APIs, meaning learn once/apply everywhere.
- I fix bugs really quickly.
- Raymond Hettinger said "The code is nicely written and is easily read start to finish." :)
- AsyncIO via [aiopeewee](#) / [muffin-peewee](#)
- Admin interface via [flask-admin](#)
- Cool [extensions](#)
- Fast, memory efficient, good performance for an ActiveRecord ORM

Weaknesses:

- Pretty much just me working on it, though I do accept patches and work hard to fix bugs quickly.
- Small ecosystem of third-party libraries / integrations
- ActiveRecord as opposed to [Data Mapper / ID Map / Unit-of-work implemented in SQLA](#)
- Lower "google-ability" factor
- Fewer stackoverflow answers
- No automatic schema migrations
- Doesn't support Oracle or Microsoft SQL Server.
- It's named peewee

Pq não SQLAlchemy?



[deleted] 4 years ago · edited 4 years ago

I'm the author of Peewee so as you might expect, I'm much more familiar with it's strengths and weaknesses. I can say, however, that SQLAlchemy is the gold standard for ORM in the Python world. It has a very active community and a maintainer who is committed to excellence. If you're a glass-half-empty guy, to put it another way, you can't go wrong if you choose SQLAlchemy.

Why would you choose Peewee, then?

- It's damn small, and easy to pick up and use. I think this is the #1 reason people use it.
- I've worked very hard to provide consistent APIs, meaning learn once/apply everywhere.
- I fix bugs really quickly.
- Raymond Hettinger said "The code is nicely written and is easily read start to finish." :)
- AsyncIO via [aiopeewee](#) / [muffin-peewee](#)
- Admin interface via [flask-admin](#)
- Cool [extensions](#)
- Fast, memory efficient, good performance for an ActiveRecord ORM

Weaknesses:

- Pretty much just me working on it, though I do accept patches and work hard to fix bugs quickly.
- Small ecosystem of third-party libraries / integrations
- ActiveRecord as opposed to [Data Mapper](#) / [ID Map](#) / [Unit-of-work implemented in SQLA](#)
- Lower "google-ability" factor
- Fewer stackoverflow answers
- No automatic schema migrations
- Doesn't support Oracle or Microsoft SQL Server.
- It's named peewee

Sou o autor do Peewee, então, como você pode esperar, estou muito mais familiarizado com seus pontos fortes e fracos. Posso dizer, no entanto, que SQLAlchemy é o padrão ouro para ORM no mundo Python. Possui uma comunidade muito ativa e um mantenedor comprometido com a excelência. Se você é um cara com o copo meio vazio, dito de outra forma, você não pode errar se escolher SQLAlchemy.

Pq não SQLAlchemy?



[deleted] 4 years ago · edited 4 years ago

I'm the author of Peewee so as you might expect, I'm much more familiar with it's strengths and weaknesses. I can say, however, that SQLAlchemy is the gold standard for ORM in the Python world. It has a very active community and a maintainer who is committed to excellence. If you're a glass-half-empty guy, to put it another way, you can't go wrong if you choose SQLAlchemy.

Why would you choose Peewee, then?

- It's damn small, and easy to pick up and use. I think this is the #1 reason people use it.
- I've worked very hard to provide consistent APIs, meaning learn once/apply everywhere.
- I fix bugs really quickly.
- Raymond Hettinger said "The code is nicely written and is easily read start to finish." :)
- AsyncIO via [aiopeewee](#) / [muffin-peewee](#)
- Admin interface via [flask-admin](#)
- Cool [extensions](#)
- Fast, memory efficient, good performance for an ActiveRecord ORM

Weaknesses:

- Pretty much just me working on it, though I do accept patches and work hard to fix bugs quickly.
- Small ecosystem of third-party libraries / integrations
- ActiveRecord as opposed to [Data Mapper](#) / [ID Map](#) / [Unit-of-work implemented in SQLA](#)
- Lower "google-ability" factor
- Fewer stackoverflow answers
- No automatic schema migrations
- Doesn't support Oracle or Microsoft SQL Server.
- It's named peewee

Por que você escolheria Peewee, então?

- É muito pequeno e fácil de pegar e usar. Acho que esta é a razão número 1 pelas quais as pessoas o usam.
- Trabalhei muito para fornecer APIs consistentes, o que significa aprender uma vez / aplicar em todos os lugares.
- Eu corrijo bugs muito rapidamente.
- Raymond Hettinger disse: "O código está bem escrito e é facilmente lido do início ao fim." :)
- AsyncIO via [aiopeewee](#) / [muffin-peewee](#)
- Interface de administração via flask-admin
- Extensões legais Rápido
- com memória eficiente, bom desempenho para um ActiveRecord ORM

Pq não SQLAlchemy?



[deleted] 4 years ago · edited 4 years ago

I'm the author of Peewee so as you might expect, I'm much more familiar with it's strengths and weaknesses. I can say, however, that SQLAlchemy is the gold standard for ORM in the Python world. It has a very active community and a maintainer who is committed to excellence. If you're a glass-half-empty guy, to put it another way, you can't go wrong if you choose SQLAlchemy.

Why would you choose Peewee, then?

- It's damn small, and easy to pick up and use. I think this is the #1 reason people use it.
- I've worked very hard to provide consistent APIs, meaning learn once/apply everywhere.
- I fix bugs really quickly.
- Raymond Hettinger said "The code is nicely written and is easily read start to finish." :)
- AsyncIO via [aiopeewee](#) / [muffin-peewee](#)
- Admin interface via [flask-admin](#)
- Cool [extensions](#)
- Fast, memory efficient, good performance for an ActiveRecord ORM

Weaknesses:

- Pretty much just me working on it, though I do accept patches and work hard to fix bugs quickly.
- Small ecosystem of third-party libraries / integrations
- ActiveRecord as opposed to [Data Mapper / ID Map / Unit-of-work implemented in SQLA](#)
- Lower "google-ability" factor
- Fewer stackoverflow answers
- No automatic schema migrations
- Doesn't support Oracle or Microsoft SQL Server.
- It's named peewee

Fraquezas:

- Praticamente só eu trabalhando nisso, embora eu aceite patches e trabalhe duro para consertar bugs rapidamente.
- Pequeno ecossistema de bibliotecas / integrações de terceiros
- ActiveRecord em oposição ao Mapeador de Dados / Mapa de ID / Unidade de trabalho implementado no SQLA
- Fator de "capacidade google" inferior
- Menos respostas stackoverflow
- Sem migrações de esquema automáticas
- Não suporta Oracle ou Microsoft SQL Server.
- Chama-se peewee

Quando devo optar pelo peewee?



- MVCs
- Microsserviços
- Projetos pequenos
- Quando você precisa de simplicidade
- O SQLAlchemy é um monstro
- FAAS

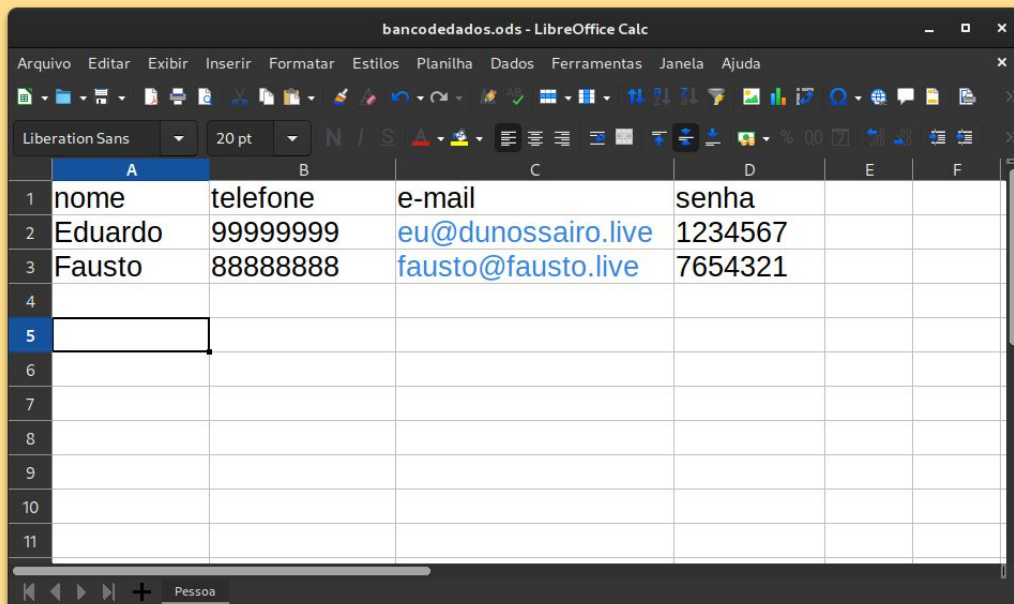
ORM

Object Relational
Mapper

Afinal, o que é ORM?



Um ORM (Object-Relational Mapper). Como funciona?



The screenshot shows a LibreOffice Calc spreadsheet with the following data:

	A	B	C	D	E	F
1	nome	telefone	e-mail	senha		
2	Eduardo	99999999	eu@dunossairo.live	1234567		
3	Fausto	88888888	fausto@fausto.live	7654321		
4						
5						
6						
7						
8						
9						
10						
11						

Afinal, o que é ORM?



Um ORM (Object-Relational Mapper). Como funciona?

bancodedados.ods - LibreOffice Calc

	A	B	C	D	E	F
1	nome	telefone	e-mail	senha		
2	Eduardo	999999999	eu@dunossairo.live	1234567		
3	Fausto	888888888	fausto@fausto.live	7654321		
4						
5						
6						
7						
8						
9						
10						
11						

Pessoa

```
1 class ...(Model):
2     nome = ...
3     telefone = ...
4     email = ...
5     senha = ...
6
7     class Meta:
8         database = ...
```

Afinal, o que é ORM?



Um ORM (Object-Relational Mapper). Como funciona?

bancodedados.ods - LibreOffice Calc

	A	B	C	D	E	F
1	nome	telefone	e-mail	senha		
2	Eduardo	999999999	eu@dunossairo.live	1234567		
3	Fausto	888888888	fausto@fausto.live	7654321		
4						
5						
6						
7						
8						
9						
10						
11						

Fields / Campos

```
1 class ...(Model):
2     { nome = ...
3       telefone = ...
4       email = ...
5       senha = ...
6
7     class Meta:
8         database = ...
```

Afinal, o que é ORM?



Um ORM (Object-Relational Mapper). Como funciona?

bancodedados.ods - LibreOffice Calc

	A	B	C	D	E	F
1	nome	telefone	e-mail	senha		
2	Eduardo	99999999	eu@dunossairo.live	1234567		
3	Fausto	88888888	fausto@fausto.live	7654321		
4						
5						
6						
7						
8						
9						
10						
11						

Table / Tabela

Pessoa

```
1 class Pessoa(Model):
2     nome = ...
3     telefone = ...
4     email = ...
5     senha = ...
6
7 class Meta:
8     database = ...
```

Afinal, o que é ORM?



Um ORM (Object-Relational Mapper). Como funciona?

bancodedados.ods - LibreOffice Calc

	A	B	C	D	E	F
1	nome	telefone	e-mail	senha		
2	Eduardo	999999999	eu@dunossairo.live	1234567		
3	Fausto	888888888	fausto@fausto.live	7654321		
4						
5						
6						
7						
8						
9						
10						
11						

Tipos / Types

```
1 class Pessoa(Model):
2     nome = CharField()
3     telefone = CharField()
4     email = CharField()
5     senha = CharField()
6
7     class Meta:
8         database = ...
```

Afinal, o que é ORM?



Um ORM (Object-Relational Mapper). Como funciona?

bancoodedados.xls - LibreOffice Calc

	A	B	C	D	E	F
1	nome	telefone	e-mail	senha		
2	Eduardo	99999999	eu@dunossairo.live	1234567		
3	Fausto	88888888	fausto@fausto.live	7654321		
4						
5						
6						
7						
8						
9						
10						
11						

Banco de dados / Database

```
1 db = SQLiteDatabase('bancoodedados.db')
2
3 class Pessoa(Model):
4     nome = CharField()
5     telefone = CharField()
6     email = CharField()
7     senha = CharField()
8
9 class Meta:
10     database = db
```

Afinal, o que é ORM?



Um ORM (Object-Relational Mapper). Como funciona?

bancodedados.ods - LibreOffice Calc

	A	B	C	D	E	F
1	nome	telefone	e-mail	senha		
2	Eduardo	99999999	eu@dunossairo.live	1234567		
3	Fausto	88888888	fausto@fausto.live	7654321		
4						
5						
6						
7						
8						
9						
10						
11						

1 eduardo = Pessoa(
2 nome='eduardo',
3 telefone='99999999',
4 email='eu@dunossairo.live',
5 senha='1234567'
6)

```
1 db = SQLiteDatabase('bancodedados.db')
2
3 class Pessoa(Model):
4     nome = CharField()
5     telefone = CharField()
6     email = CharField()
7     senha = CharField()
8
9     class Meta:
10         database = db
```

pee

wee

De volta

Nosso projetinho



Nosso objetivo hoje é criar um sisteminha de notas

Cada **Pessoa** vai poder criar **Grupos** de **Notas**

Vamos precisar criar 3 tabelas:

- Pessoa
- Grupo
- Notas



Uma visão geral



Grupo	
Dona	
Nome	Text

Pessoa	
Nome	Text
Email	Text
Senha	Text

Nota	
Grupo	
Dona	
Título	Text
Nota	Text
Criado em	Datetime
Modificado em	Datetime

Lets que bora



Bora criar essas tabelas



Criando tabelas



Existem 3 formas de criar tabelas com Peewee:

- Tabela
 - `Table.create_table()`
- db
 - `db.create_tables([A, B, C])`
- Migração
 - `playhouse`
 - `from playhouse import migrate, SchemaMigrator`

Inserindo dados



Para inserir dados no peewee podemos usar algumas formas.

- `dado = Tabela(dados=dados)`
 - `dado.save()`
- `Tabela.create(dados=dados)`
 - Já salva
- `BulkInsert`

Bora ver isso funcionando



```
1  p1 = Pessoa(  
2      nome='eduardo',  
3      email='eu@dunossauro.live',  
4      senha='1234567'  
5  )  
6  
7  p1.save()
```

```
1  p2 = Pessoa.create(  
2      nome='fausto',  
3      email='fausto@live.live',  
4      senha='1234567'  
5  )
```

Bulk insert



Opção para inserir muitos registros de uma vez só

```
1 registros = [  
2     {'nome': 'Irmão do jorel', 'email': 'irmao@live', 'senha': '1234567'},  
3     {'nome': 'Gesonel', 'email': 'gesinho@live.live', 'senha': '1234567'},  
4     {'nome': 'Lara', 'email': 'lara@live.live', 'senha': '1234567'},  
5 ]  
6  
7 Pessoa.insert_many(registros).execute()
```

PWIZ

Sei que não tava
no roteiro, mas é
importante

PWIZ



PWIZ é um script MUITO MANEIRO que vem na **playhouse**, que facilita a migração para seu primeiro ORM.

Imagina que você já tem um banco operando, você não precisa mais escrever as tabelas, o PWIZ gera o molde pra você <3

Exemplo de uso



```
1  python -m pwiz -e sqlite sua_base.db
2
3  python -m pwiz -e mysql -s user -P senha -p porta seu_mysql
4
5  python -m pwiz -e postgresql -s user -P senha -p porta seu_postgres
```

Buscando coisas

Quer
ies

Queries



Query é uma operação para procurar registros no banco de dados. Por exemplo, imagine que queremos encontrar todas as pessoas que se chamam 'Eduardo' na nossa base de dados:

```
1 Pessoa.select().where(Pessoa.nome == 'Eduardo')
```

Queries



Query é uma operação para procurar registros no banco de dados. Por exemplo, imagine que queremos encontrar todas as pessoas que se chamam 'Eduardo' na nossa base de dados:

```
1 Pessoa.select().where(Pessoa.nome == 'Eduardo')
```

Queries



Query é uma operação para procurar registros no banco de dados. Por exemplo, imagine que queremos encontrar todas as pessoas que se chamam 'Eduardo' no nosso banco de dados:

```
1 Pessoa.select().where(Pessoa.nome == 'Eduardo')
```

The diagram shows a code editor window with the query `1 Pessoa.select().where(Pessoa.nome == 'Eduardo')`. Three callout boxes are present: 'o que?' points to `select()`, 'Filtro' points to `.where(Pessoa.nome == 'Eduardo')`, and 'Tabela' points to `Pessoa`.

O que queremos?



Uma forma de otimizar as buscas é dizer o que queremos exatamente. Para isso vamos usar o método de **select**. Vamos imaginar que queremos email e senha de todos os eduardos na base

```
1 emails_senhas = (  
2     Pessoa  
3     .select(Pessoa.email, Pessoa.senha)  
4     .where(Pessoa.nome == 'Eduardo')  
5 )
```

O que queremos?



Uma forma de otimizar as buscas é dizer o que queremos exatamente. Para isso vamos usar o método de **select**. Vamos imaginar que queremos email e senha de todos os eduardos na base

```
1 emails_senhas = (  
2     Pessoa  
3     .select(Pessoa.email, Pessoa.senha)  
4     .where(Pessoa.nome == 'Eduardo')  
5 )
```

Juntando coisas (join)



Imagine que você precisa pegar somente as notas que fazem parte de um grupo X, como faríamos isso?

```
1  notas_x = (  
2      Nota.select()  
3      .join(Grupo)  
4      .where(Grupo.nome == 'x')  
5  )
```


Juntando coisas (join)



Imagine que você precisa pegar somente as notas que fazem parte de um grupo X, como faríamos isso?

```
1 notas_x = (  
2     Nota.select()  
3     .join(Grupo)  
4     .where(Grupo.nome == 'x')  
5 )
```

Notas do grupo X

Operadores de Query



Bom, até agora usamos somente `==`, porém existem muitos outros operadores

- `!=`
 - `Pessoa.nome != 'Charles'`
- `<`, `>`, `<=`, `>=`
 - `Pessoa.idade > 18`
 - `Pessoa.idade < 18`

Operadores de Query



- `in_`
 - `Pessoa.select().where(Pessoa.nome.in_(['eduardo', 'Fausto']))`
 - Retorna eduardos e faustos
- `not_in`
 - `Pessoa.select().where(Pessoa.nome.not_in(['eduardo', 'Fausto']))`
 - Retorna não eduardos e faustos
- `contains`
 - `Pessoa.select().where(Pessoa.nome.contains('a'))`
 - Retorna todas as pessoas que têm **a** no nome

Combinadores de query



Embora tenhamos muitas opções de operadores, as vezes eles não são o suficiente para uma query mais elaborada.

Imagine um cenário: Você precisa enviar um e-mail para todas as pessoas em que o nome começa com **A** e são **Maiores de 18**

```
1  (Pessoas
2    .select(Pessoa.email)
3    .where(Pessoa.nome.startswith('a') & Pessoa.idade >= 18)
4  )
```

Combinadores de query



Existem 3 tipos de combinadores:

- &
 - junta wheres
 - A e B
- |
 - A ou B
- ~
 - Negação
 - `where(~(Pessoa.idade > 18))`
 - Somente maiores de idade

Alterando dados

Após selecionarmos o objeto que queremos alterar, podemos simplesmente atribuir um novo valor a esse campo:

```
1  nota = (  
2      Nota  
3      .select()  
4      .where(Nota.title == 'Batatinha')  
5      .get()  
6  )  
7  
8  nota.title = 'xpto'  
9  nota.save()
```

Deletando dados

```
1  ( # deletar tudo
2    Nota
3    .delete()
4    .where(Nota.title == 'Batatinha')
5  )
6
7  nota.delete() # deleta 1
```

Você pode usar `.delete()` da maneira que achar mais conveniente. Em uma query ou em um elemento específico.

Estendendo o
Peewee

Play
House

Playhouse



Caso o recurso que você queira não esteja na biblioteca padrão, podemos usar o playhouse.

- Testes
- Fields customizados
- Funções
- DataSets
- Reflexão (PWIZ)
- Migrações



picpay.me/dunossauro



apoia.se/livedepython



PIX



Ajude o projeto

