Anvil ORM

Owen Campbell

Jun 11, 2021

CONTENTS

For this tutorial, we're going to build a model to represent Books and their Authors. You will need:

- A new anvil app
- The data tables service enabled
- This model library installed

For a book, we want to store its title, publication date and author. For an author, we want to store a first name and a last name.

CHAPTER

ONE

INSTALL THE ORM

Create a package in the server code section of your app named 'orm_server' and two modules within that package named 'persistence' and 'security'. Replace the contents of both modules with the code from persistence.py and security.py respectively.

Create a package named 'orm_client' in the client code section of your app and a module within that package named 'particles'. Replace its content with the code from particles.py

CHAPTER

TWO

CREATE THE DATA TABLES

Create a data table called ‘author’ with three columns:

- a number column called ‘uid’
- a text column called ‘first_name’
- a text column called ‘last_name’

Create a second table called ‘book’ with four columns:

- a number column called ‘uid’
- a text column called ‘title’
- a date column called ‘published_on’
- a column which links to a single column in the ‘author’ table named ‘author’.

CHAPTER

THREE

CREATE THE MODEL MODULE

Create a module in the client code section of your app and call it 'model'. Add the following code to define the Book and Author classes:

```
from .orm_client.particles import model_type, Attribute, Relationship

@model_type
class Author:
    first_name = Attribute()
    last_name = Attribute()


@model_type
class Book:
    title = Attribute()
    published_on = Attribute()
    author = Relationship(cls="Author")
```

CHAPTER

FOUR

ADD SOME ENTRIES

Create a new module in the client code section of your app, name it 'startup' and set it as the startup module. Delete its content and replace it with:

```
import datetime as dt
from .model import Book, Author

print("Creating Luciano")
luciano = Author(first_name="Luciano", last_name="Ramalho").save()

print("Creating Drew")
drew = Author(first_name="Drew", last_name="Neil").save()

print("Creating Fluent Python")
fluent_python = Book(
    title="Fluent Python",
    published_on=dt.datetime(2015, 8, 21).date(),
    author=luciano
).save()

print("Creating Practical Vim")
practical_vim = Book(
    title="Practical Vim",
    published_on=dt.datetime(2017, 1, 1).date(),
    author=drew
).save()
```

Launch your app and stop it again. You should see the results of the print statements in its output console.

Check the contents of your data tables. You should see both authors and books in their respective tables, with id numbers automatically assigned. You should also see two entries in the sequence table with the 'next' values ready for the next author and book.

CHAPTER

FIVE

FETCH THE ENTRIES BACK AGAIN

Delete the code in your startup app (or rename it and create a new 'startup' module) and replace the content with:

```
from .model import Book, Author

for book in Book.search():
    print(book.title)
    print(book.author.first_name)
```

Launch the app and you should see the title and author's first name for both books in your output consule.

CHAPTER

SIX

MAKE A CHANGE

Delete the code in your startup app (or rename it and create a new 'startup' module) and replace the content with:

```
from .model import Book, Author

fluent_python = Book.search(title="Fluent Python")[0]
fluent_python.title = "Fluent Python (Clear, Concise, and Effective Programming)"
fluent_python.save()

practical_vim = Book.get(id=2)
practical_vim.title = "Practical Vim (Edit Text at the Speed of Thought)"
practical_vim.save()
```

Start and stop the app and check your data tables. You should see the updated titles for both book rows.

CHAPTER

SEVEN

HOWTO...

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search