Flask Tutorial – Flask SQLAlchemy with MySQL

by Parwiz

In this **Flask Tutorial** we are going to learn about **Flask SQLAlchemy with MySQL** database, i will show you that how you can connect your Flask application to Mysql database using

Flask SQLAlchemy.

Also if you are interested in Django, you can check this link Django Web Development Tutorials.

What is SQLAlchemy ?

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application

developers the full power and flexibility of SQL.

It provides a full suite of well known enterprise-level persistence patterns, designed for

efficient and high-performing database access, adapted into a simple and Pythonic

domain language.

What is Flask-SQLAlchemy ?

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks.

See the SQLAlchemy documentation to learn how to work with the ORM in depth.

The following documentation is a brief overview of the most common tasks, as well as the features specific to Flask-SQLAlchemy.

Installation

You can simply install Flask-SQLAlchemy using pip command.

```
pip install Flask-SQLAlchemy
```

So now after installation you need to create database, we are using Mysql database. for this you need to download and install Wamp Server. create a database in the Wamp Server, iam going to give *flaskcodeloop* for the database name, but you can give the name according to your choice.

right now we don’t have any table in our database, we will create this using Flask SQLAlchemy.

We are going to use codes from our previous articles.

This is our *app.py* and we have added our SQLAlchmey and database configuration.

```
from flask import Flask, render_template, flash, request, redirect, url_for
from forms import LoginForm
from flask_sqlalchemy import SQLAlchemy
```

```
#create the object of Flask
app  = Flask(__name__)

app.config['SECRET_KEY'] = 'hardsecretkey'


#SqlAlchemy Database Configuration With Mysql
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:''@localhost/flaskcodeloop'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False



db = SQLAlchemy(app)


#our model
class UserInfo(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    username = db.Column(db.String(100), unique = True)
    password = db.Column(db.String(100))


    def __init__(self, username, password):
        self.username = username
        self.password = password







#creating our routes
@app.route('/')
def index():

    return render_template('index.html')



#login route
@app.route('/login' , methods = ['GET', 'POST'])
def Login():
    form = LoginForm()

    if form.validate_on_submit():
        if request.form['username'] != 'codeloop' or request.form['password'] !=
            flash("Invalid Credentials, Please Try Again")


        else:
            return redirect(url_for('index'))



    return render_template('login.html', form = form)





#run flask app
if __name__ == "__main__":
    app.run(debug=True)
```

In the above code this is for SQLAlchemy.

```
#SqlAlchemy Database Configuration With Mysql
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:''@localhost/flaskcodeloc
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False


db = SQLAlchemy(app)
```

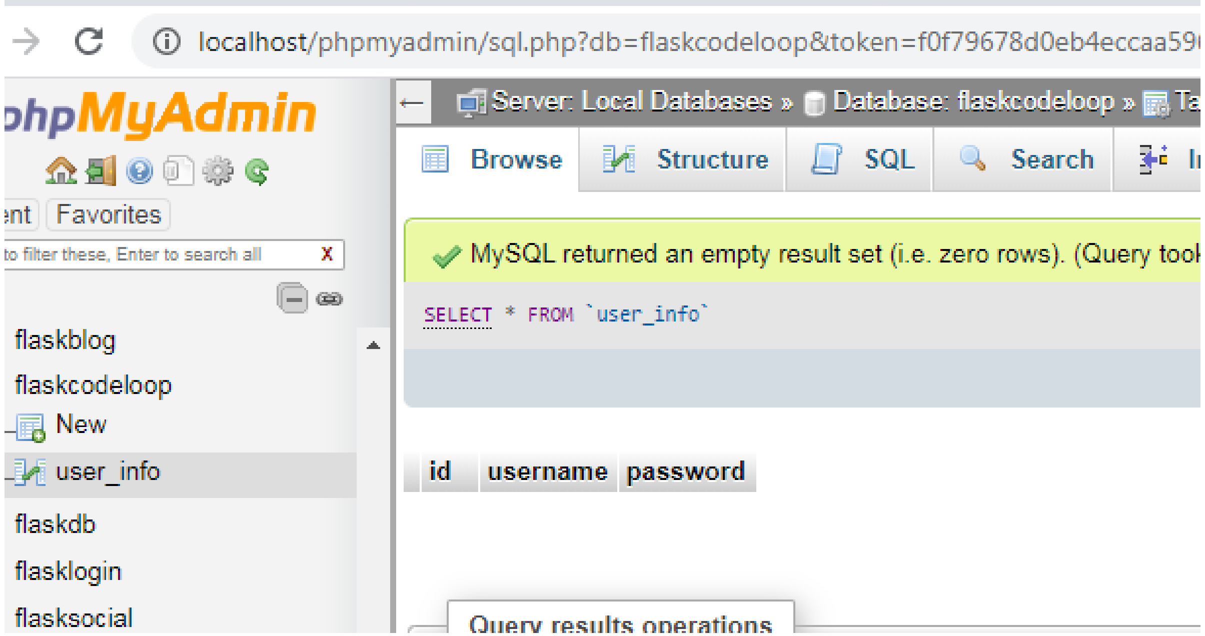
And this is our database model class, it is just a simple model with three fields, id, username and password.

```
class UserInfo(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    username = db.Column(db.String(100), unique = True)
    password = db.Column(db.String(100))


    def __init__(self, username, password):
        self.username = username
        self.password = password
```

We have created our database model in our *app.py*, now we need to add this model in our Mysql Database, for this you need to open your Python terminal in Pycharm IDE. first we need to import our db from app, and after that we create our tables in the database.

```
>>> from app import db
>>> db.create_all()
```

If you check your *flaskcodeloop* database, you can see that we have our table with the data.



localhost/phpmyadmin/sql.php?db=flaskcodeloop&token=f0f79678d0eb4eccaa59

✓ MySQL returned an empty result set (i.e. zero rows). (Query took

```
SELECT * FROM `user_info`
```

id	username	password

Flask Tutorial – Flask SQLAlchemy with MySQL

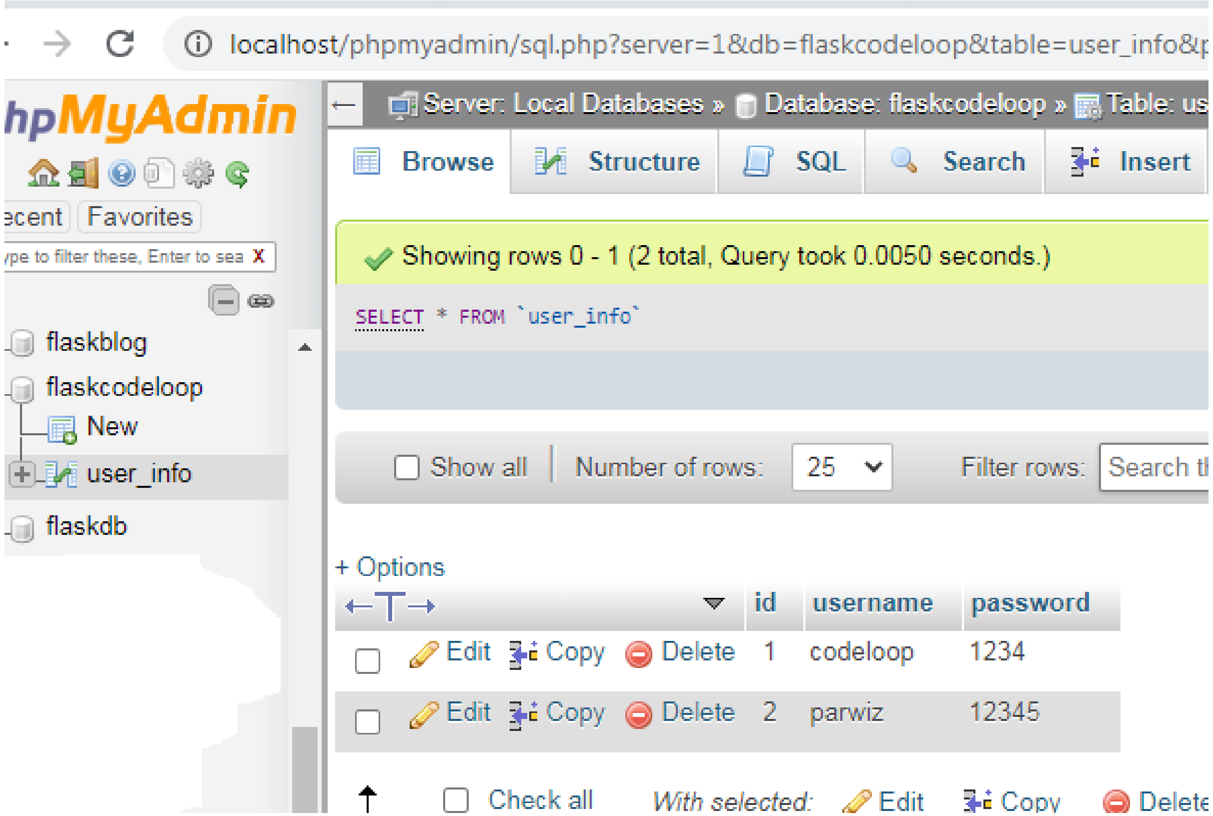
Now let’s add some data to our database. basically we are going to add just two

UserInfo

to our database table.

```
>>> from app import db
>>> from app import UserInfo
>>> u = UserInfo('codeloop',1234)
>>> u1 = UserInfo('parwiz', 12345)
>>> db.session.add(u)
>>> db.session.add(u1)
>>> db.session.commit()
```

Check your database table, you will have two *UserInfo* data.



✓ Showing rows 0 - 1 (2 total, Query took 0.0050 seconds.)

```
SELECT * FROM `user_info`
```

	id	username	password
□ Edit Copy Delete	1	codeloop	1234
□ Edit Copy Delete	2	parwiz	12345

Flask Tutorial – Flask SQLAlchemy with Mysql9n

Selecting Data

Also you can select or retrieve the data from your Mysql Database, so now let’s open our

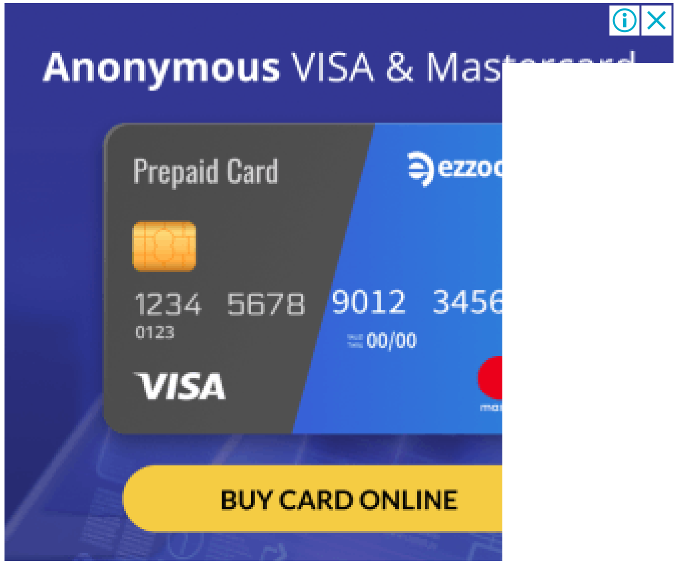
python terminal in Pycharm IDE, and retrieve our data.

```
>>> user_data = UserInfo.query.all()
>>> for user in user_data:
...     print(user.username)
...
codeloop
parwiz
```

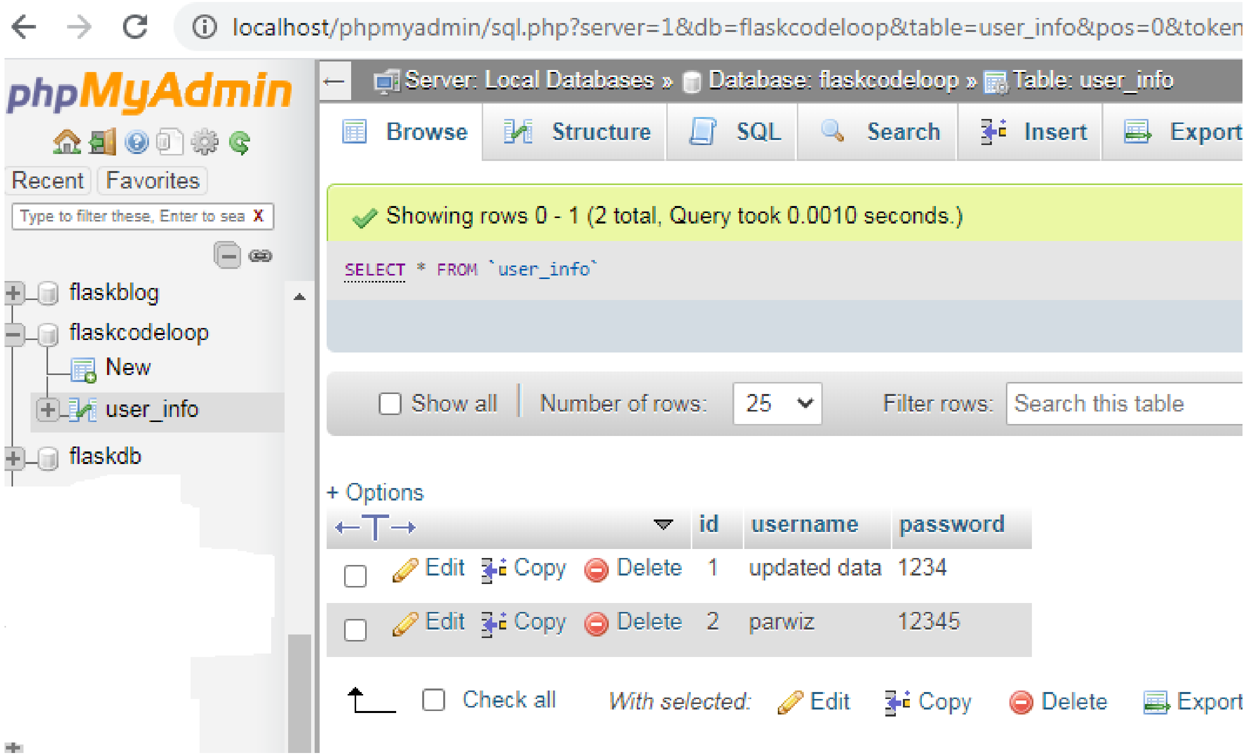
Updating Data

updating data is also an easy task with Flask SQLAlchemy, let’s update our first record in the database.

```
>>> update = UserInfo.query.filter_by(id=1).first()
>>> update.username = 'updated data'
>>> db.session.commit()
```



Now check your Mysql database, the first record is updated.



localhost/phpmyadmin/sql.php?server=1&db=flaskcodeloop&table=user_info&pos=0&token

Server: Local Databases » Database: flaskcodeloop » Table: user_info

✓ Showing rows 0 - 1 (2 total, Query took 0.0010 seconds.)

```
SELECT * FROM `user_info`
```

□ Show all | Number of rows: 25 | Filter rows: Search this table

id	username	password
1	updated data	1234
2	parwiz	12345

Flask SQLAlchemy Updating Data

Deleting Data

Also you can delete data using this commands in Flask SQLAlchemy.

```
>>> deleted = UserInfo.query.filter_by(id=1).first()
>>> db.session.delete(deleted)
```

OK now we have done some basic operations in our database using Flask SQLAlchemy,

it was just using python terminal, we will do that practically in the next articles. also these

are the html files for the *app.py* file.

This is our *base.html,* we have already talked about creating templates in Flask, you can

read this article Introduction to Flask Templates. you need to just

create *templates* folder

in your working directory.

templates/base.html

```
<!DOCTYPE html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %} {% endblock %}</title>

    <!-- CSS Bootstrap CDN Link -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/c

</head>
<body>



<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">

      <a class="navbar-brand" href="{{url_for('index')}}">CodeLoop</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="{{url_for('index')}}">Home <span class="sr-only
      </li>



         <li class="nav-item">
        <a class="nav-link" href="#">Logout</a>
      </li>


    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aric
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</
    </form>
  </div>

  </div>

    <a href="{{url_for('Login')}}"><button class="btn btn-success navbar-btn">Log
     <a href=""><button class="btn btn-success navbar-btn">Signup</button> </a>
</nav>
```

```
<!-- JS, Popper.js, and jQuery -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.


{% block body %}


{% endblock %}




</body>
</html>
```

our *login.html* with flash message.

templates/login.html

```
{% extends 'base.html' %}


{% block title %} Home {% endblock %}

{% block body %}

<div class="container">
```

```
    <h1>Home Page - Welcome to codeloop.org</h1>
<h3>Tutorial Number 11 </h3>

    <br>
    <br>
    <hr>

    {% with messages = get_flashed_messages() %}

    {% if messages %}

    {% for message in messages %}

       <div class="alert alert-success alert-dismissable" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="clos
            <span aria-hidden="true">X</span>
        </button>

             {{message}}

    </div>




    {% endfor  %}

    {% endif %}

    {% endwith %}

    <h1>Please Login</h1>

    <form action="" method="post" novalidate>
      {{form.csrf_token}}

      <p>

          {{form.username.label}}
          {{form.username(size=32)}}



          {% for error in form.username.errors %}
          <span style="color:red;">

              {{error}}
          </span>
          {% endfor %}
          {% endfor %}

      </p>


         <p>

          {{form.password.label}}
          {{form.password(size=32)}}


            {% for error in form.password.errors %}
          <span style="color:red;">

              {{error}}
          </span>
          {% endfor %}

      </p>


        <p>
            <input type="submit" value="Login" class="btn btn-success">

        </p>



    </form>



</div>

{% endblock %}
```

And this is our *index.html* file.

templates/index.html

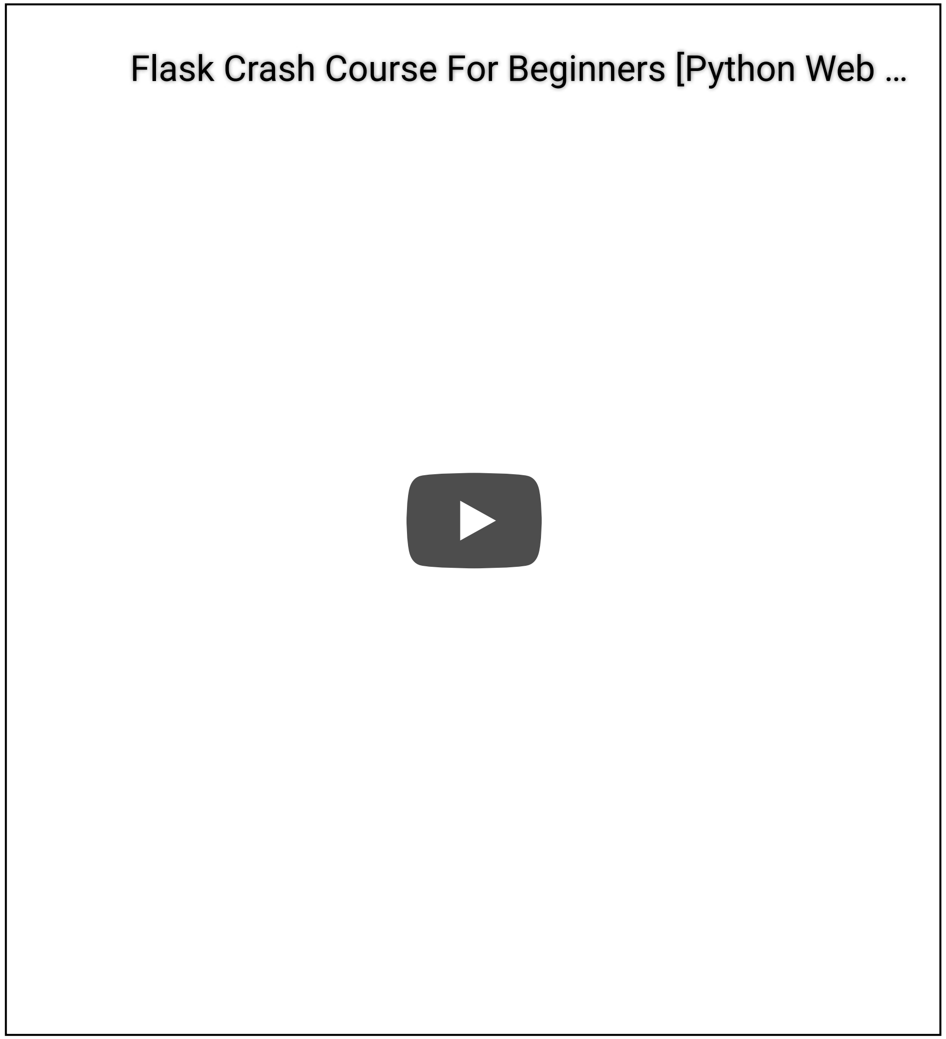
```
{% extends 'base.html' %}


{% block title %} Home {% endblock %}

{% block body %}

<div class="container">
    <h1>Home Page - Welcome to codeloop.org</h1>
<h3>Tutorial Number 11 </h3>

<p>
In this tutorial we are going to talk
about Flask SQLAlchemy.

</p>




</div>

{% endblock %}
```

This is *forms.py* file.

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField
from wtforms.validators import InputRequired
```

```
class LoginForm(FlaskForm):
    username = StringField('Username', validators=[InputRequired()])
    password = PasswordField('Password', validators=[InputRequired()])
```

Also you can watch my complete 4 hours training on Flask Web Development.



Flask Crash Course For Beginners [Python Web …

Thank you for visiting my website! If you enjoyed this article, please consider supporting my works on Patreon.

BECOME A PATRON

Flask CRUD Application with SQLAlchemy Codeloop

codeloop.org

Flask Tutorial - Custom Error Page in Flask Codeloop

codeloop.org

Flask Archives – Codeloop

codeloop.org

How to Build REST API with SQLAlchemy & Marshmallow Codeloop

codeloop.org

Flask Tutoria
Forms with F
WTF Codelo

codeloop.org

Parwiz Forogh

YouTube

Like 554 people like this. Sign Up to see what your friends like.

Flask

Flask, Flask Web Development, Python Web Development

‹ Flask Tutorial – Creating Flask Flash Messages

› Java GUI – JavaFX Shape Drawing Line

1 thought on “Flask Tutorial – Flask SQLAlchemy with MySQL”

client

at

tanQ very very good

Reply

Leave a Comment

Name *

Email *

Website

Post Comment

Programming Books