AULA 12

LINGUAGEM DE PROGRAMAÇÃO II - LP2I2

INTEGRANDO COM BANCO DE DADOS

PROFA. ANA PAULA MÜLLER GIANCOLI

Instituto Federal de São Paulo - IFSP

Campus Bragança Paulista

AGENDA

- Configuração
- Executando Exemplo
- Atividades para casa

CONFIGURAÇÃO

CONFIGURAÇÃO

- Manipulando dados de um banco de dados.
- Vamos utilizar:
  - Linguagem de programação: python
  - Banco de dados: mysql
  - Conector Python: específico para a versão do seu BD com python.

CONFIGURAÇÃO

- Criaremos um banco de dados chamado lp2i2.
- Criaremos duas tabelas: aluno e alunopessoal.
- Inserir dados: utilize o arquivo fornecido.
- Utilizaremos um **cursor** de banco de dados, o qual **é** uma estrutura de controle que permite percorrer sobre os registros em um banco de dados.

CONFIGURAÇÃO

- Utilizaremos consultas parametrizadas ou instruções preparadas para acessar os dados.

- Uma consulta parametrizada é uma consulta na qual os espaços reservados são usados para parâmetros e os valores dos parâmetros fornecidos no tempo de execução. Isso significa que a consulta parametrizada é compilada apenas uma vez.

- Utilizamos variáveis Python na posição dos espaços reservados %s.

PORQUE UTILIZAR?

- Porque utilizar prepared statement e parameterized query?
  - Melhora velocidade.
  - A consulta parametrizada do MySQL é compilada apenas uma vez, podendo executar as instruções sem recompilar.
  - Mesma operação com dados diversos.
  - Impede injection attacks.

EXECUTANDO EXEMPLO

aluno.py e my_aluno.py

SIGNIFICADO

```
import mysql.connector: importa o módulo mysql.connector

from mysql.connector import Error, connect: importa a função connect e error do módulo
                                            mysql.connector

connection = mysql.connector.connect(): cria uma variável chamada connection com todos
                                        os parâmetros de conexão utilizados pelo nosso
                                        módulo. Importante: verifique se os parâmetros
                                        de conexão correspondem aos da sua máquina de
                                        desenvolvimento.

print("Tentando a conexão..."): imprime uma mensagem simples no terminal informando ao
                                usuário que o aplicativo está tentando se conectar ao
                                banco de dados.

if connection.is_connected(): verifica se o valor do método é is_connected() = True,
                              em seguida, imprime uma mensagem no terminal para informar
                              ao usuário que a conexão foi bem sucedida.

info = connection.get_server_info(): exibe as informações sobre o banco de dados, versão.

def display_all_alunos(): define uma função que executa uma consulta simples em nosso
                          banco de dados e usa um for para imprimir os resultados no
                          terminal.

cursor = connection.cursor(): cria um objeto cursor a partir da nossa variável connection.

cursor.execute("SELECT * FROM alunos;"): executa a consulta no banco de dados.
```

SIGNIFICADO

```
records = cursor.fetchall(): exibe todas as linhas retornadas em uma variável chamada records.

for row in records: efetua uma varredura na lista records, passando por todos os valores.

cursor.close(): fecha o cursor.

connection.close(): encerra a conexão com o banco de dados.

def search_alunos(query): define uma função que executa uma busca simples em nosso
                          banco de dados de acordo com o termo como parâmetro passado para
                          pesquisar os resultados e exibir no terminal.

sql_stmt = f"SELECT * FROM alunos WHERE nomecompleto LIKE '%{query}%' OR prontuario
           LIKE '%{query}%'": a instrução SELECT usa caracteres curinga para procurar
                              nomes de alunos ou número de prontuários que contenham o
                              que foi enviado no query parâmetro.

cursor.execute(sql_stmt): executa a pesquisa de acordo com o statement preparado e busca
                          no banco de dados.

def add_new_alunos(aluno): define uma função que executa uma inserção em nosso
                           banco de dados e usa uma tupla ou lista como parâmetro para
                           inserir os resultados no banco de dados.
```

SIGNIFICADO

```
cursor = connection.cursor(prepared=True): Permite que você tenha um cursor específico no
                                           qual as instruções são preparadas. E retorna uma
                                           instância de classe MySQLCursorPrepared.

cursor.execute(sql_stmt, aluno): chama o método execute do objeto cursor que aceita dois
                                 parâmetros. O primeiro parâmetro é a instrução SQL,
                                 enquanto o segundo parâmetro representa os valores reais
                                 que serão substituídos pelos espaços reservados em nossa
                                 instrução.

connection.commit(): chama o método commit do objeto connection. Esse método executa a
                     consulta no banco de dados.

cursor.execute(sql_stmt, (param,)): chama o método execute do objeto cursor que aceita dois
                                    parâmetros. O primeiro parâmetro é a instrução SQL,
                                    enquanto o segundo parâmetro representa a tupla com os
                                    ids que serão substituídos nos espaços reservados
                                    em nossa instrução.
```

PERGUNTAS ?

ATIVIDADE PARA CASA

ATIVIDADE PARA CASA

- Refatorar o código: criando uma classe conexão, uma classe aluno, uma classe alunopessoal, um arquivo que utiliza todas as classes permitindo repetição e manipulação de todas as informações.

- Implementar o crud para a classe alunopessoal similar a classe aluno.

- Entregar no link apropriado no moodle.

REFERÊNCIAS

Consulte também: https://pynative.com/python-mysql-execute-parameterized-query-using-prepared-statement/