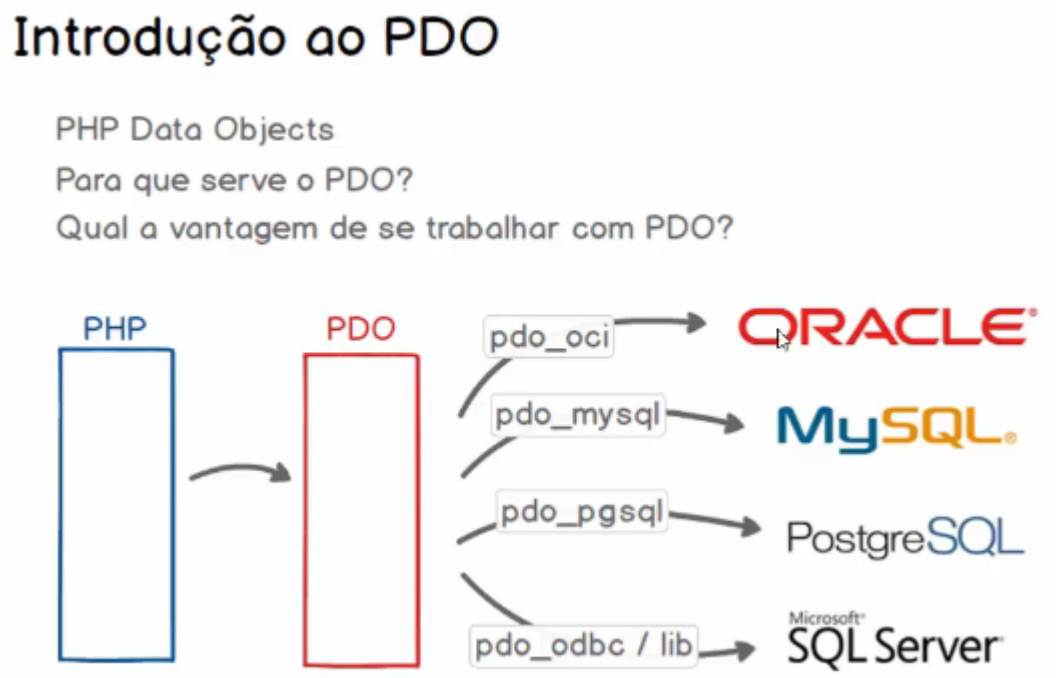
Inicio PHP - PDO

#Utilidade_PDO

O **PDO** funciona como uma interface com regras e padronizações a serem seguidas para trabalhar com diferentes tipos de bancos de dados. Caso haja a necessidade de mudar o **SGBD**, não é preciso uma refatoração total do código, similar à lógica de funções ("functions") para reaproveitamento de código.



#PHPeMySQL

Ao criar a variável de conexão, é preciso preencher algumas informações importantes que o PDO necessita. Como o PDO já é uma extensão integrada ao PHP, ele possui uma sintaxe específica para a configuração da conexão.

```
<?php

$dsn='mysql:host=localhost;dbname=php_2com_pdo';

$usuario = 'root';

$senha = '';
```

```
try{
$conexao = new PDO($dsn, $usuario, $senha);

}catch(PDOException $e){

//Tratativa  para visualizar o erro ou a Array da conexão
    // echo '<pre>';
    // print_r($e);
    // echo '</pre>';
}
?>
```

1. Data Source Name (DSN):
Fonte de dados, utilizando os prefixos dos drivers que serão utilizados. Nesse caso, é "**mysql**".
2. Host: localhost
Para máquinas pessoais utilizando XAMPP se utiliza o localhost, podendo mudar essa informação de acordo com o host utilizado (**endereço IP ou URL**) para acesso.
3. DBname (nome do banco):
Você precisa colocar exatamente o nome do banco de dados que será utilizado para a conexão. Por exemplo, "**dbname=projeto_x_2**".
4. Usuário do banco que tem acesso e senha:
No caso padrão e local, seria "**root**" e sem senha, mas pode mudar de acordo com as necessidades, permissões e outros casos dentro do projeto/empresa e db.

#Instruções

As instruções iniciais para utilização e testes da conexão tem suas diferenças e especificações e sendo elas nativas do PHP:

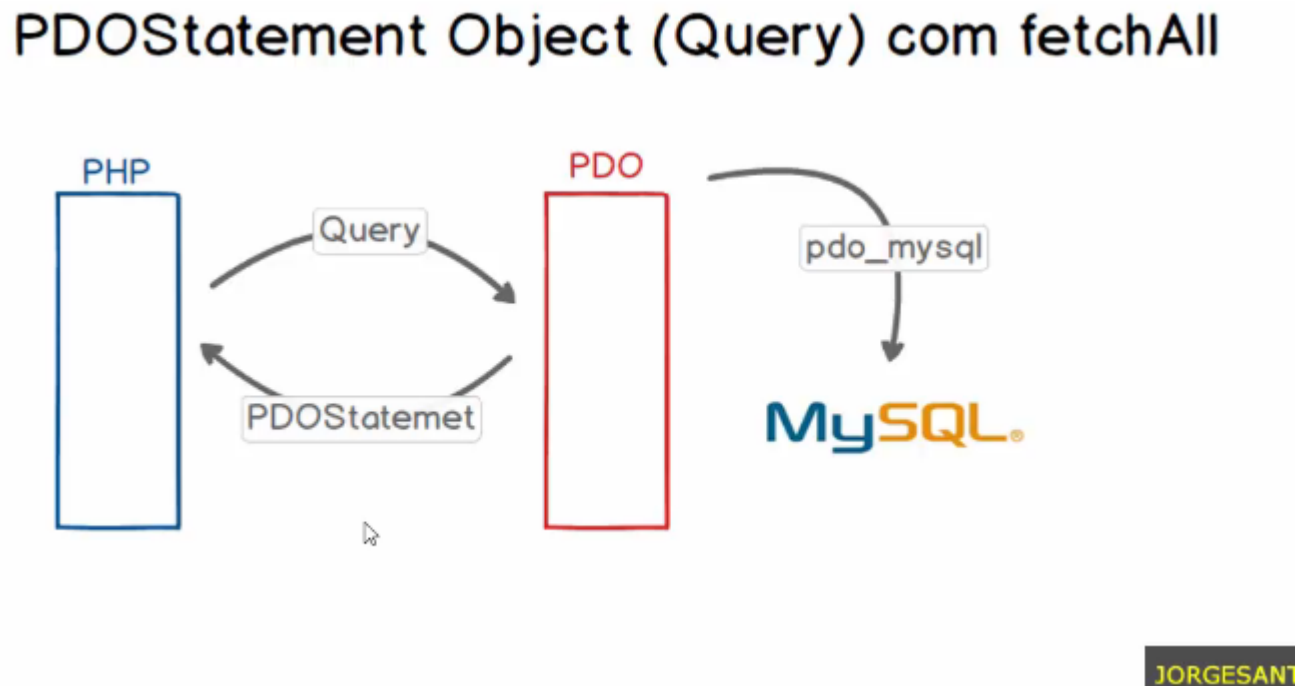
```
$conexao = new PDO($dsn, $usuario, $senha);
       $query='
        CREATE TABLE tb_usuarios(
            id int(5) primary key not null auto_increment,
            nome varchar(50),
            email varchar(50),
            senha varchar(50)
            )
    ';
```

```
$retorno = $conexao->exec($query) ;
echo $retorno;

$query_insert = '
insert into tb_usuarios(nome,email,senha) values (
    "Nicolas2",
    "nicolas2@gmail.com",
    "1234567"
    )
';

$retorno_insert = $conexao->exec($query_insert);
echo $retorno_insert;
```

O **exec** tem como principal funcionalidade executar uma query, retornando o número de linhas modificadas ou removidas. No caso do método exec, para as query do tipo SELECT ou CREATE, o retorno será 0. Já para as query do tipo UPDATE, DELETE ou INSERT, o retorno será o número de linhas modificadas ou removidas.

#PDO_Statement_Object

Conceito focado em enviar as **queries SQL** do PHP para o **PDO**, retornando um **PDOStatement**, ou seja, um objeto que contenha a declaração da query e permita recuperar as informações contidas no banco de dados.



Métodos do PDOStatemet:

- **fetchAll();** retorna todos os dados da consulta contina na $query.
  -OBS: **fechAll** por padrão trás os 2 tipos de resultado da **array(associativo e numérico)**.

  fetchAll(PDO::FETCH_ASSOC); para trazer os resultados associativos;

  fetchAll(PDO::FETCH_NUM); para os resultados numéricos;

  fetchAll(PDO::FETCH_BOTH); trazendo ambos;

  fetchAll(PDO::FETCH_OBJ); Mudando array para um object e classe (echo $lista[1]->nome;)

```
//METODO PADRÃO VAI TRAZER OS 2 TIPOS DE ARRAY
$conexao = new PDO($dsn, $usuario, $senha);

$query = '
SELECT * FROM tb_usuarios;
';

$stmt = $conexao->query($query);
$lista = $stmt ->fetchAll();

echo '<pre>';
```

```
print_r($lista);
echo '</pre>';

echo $lista[0]['nome'];
echo '<br />';
echo $lista[0][1];
```

- **fetch();** retorna somente 1 resultado, utilizado para evitar a estruturação da array principal do banco de dados, que seria a array de ID's;
  -OBS: Também é possível utilizar PDO::FETCH_ASSOC / NUM / OBJ / BOTH

```
$query = '
SELECT * FROM tb_usuarios where id = 1;
';

$stmt = $conexao->query($query);
$usuario = $stmt ->fetch(PDO::FETCH_ASSOC);

echo '<pre>';
print_r($usuario);
echo '</pre>';

echo $usuario['nome'];
```

- Não é exatamente um método mas um modo de listar as informações utilizando foreach: Utilizando o foreach direto na conexão

```
$conexao = new PDO($dsn, $usuario, $senha);
$query = '
SELECT * FROM tb_usuarios;
';

foreach($lista_usuario as $key => $value){
     echo $value['nome'];
     echo '<hr>';
 }
```

ou utilizando no select no método **PDOStatemet na variável $lista_usuario**:

```
$conexao = new PDO($dsn, $usuario, $senha);
$query = '
SELECT * FROM tb_usuarios;
';

$stmt = $conexao->query($query);
$lista_usuario = $stmt ->fetchAll(PDO::FETCH_ASSOC);

foreach($lista_usuario as $key => $value){
    echo $value['nome'];
    echo '<hr>';

}
```

#codigo_explicativo

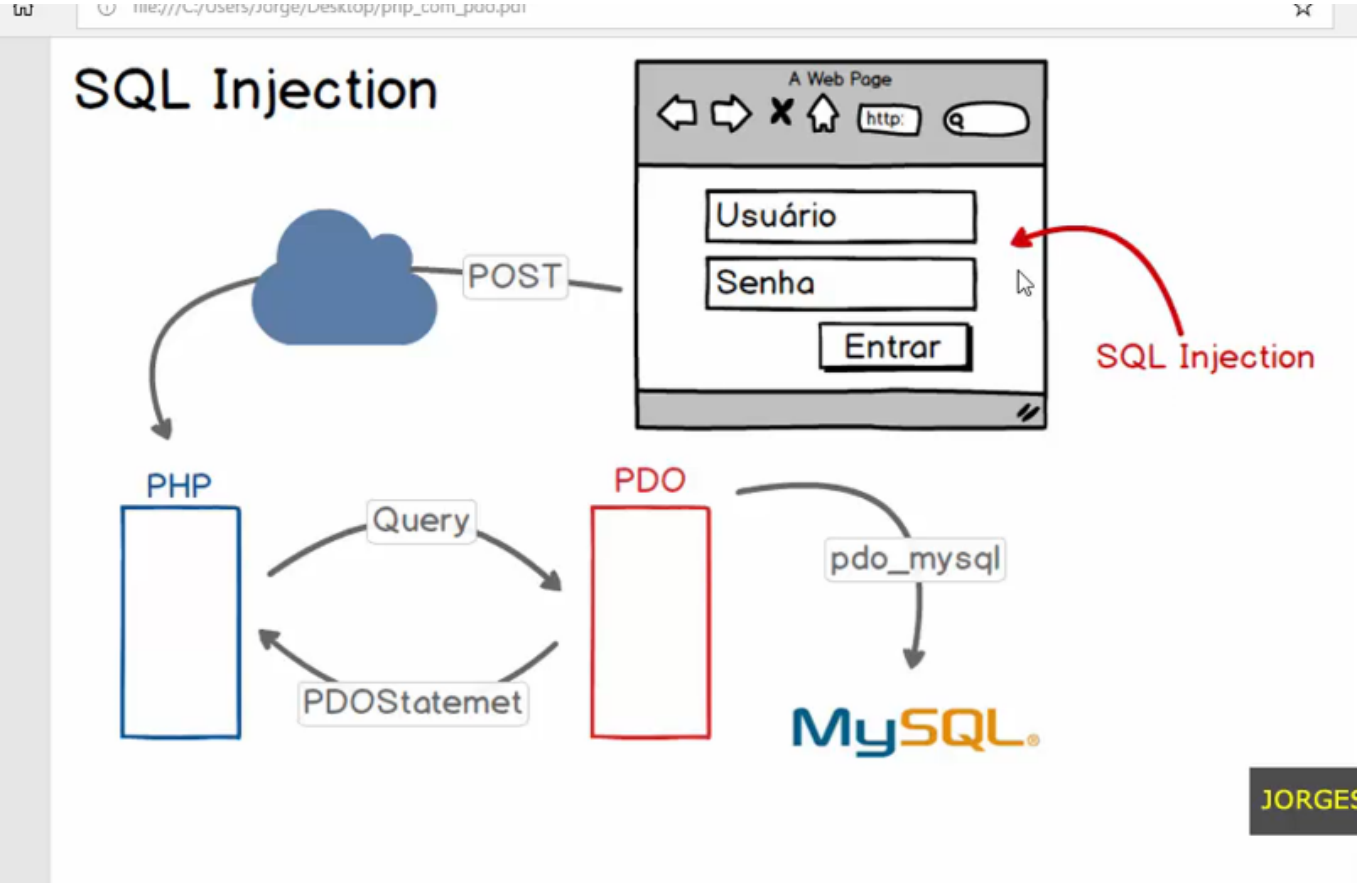
```
<?php
// print_r($_POST);
//VERIFICAÇÃO SE OS CAMPOS ESTÁ PREENCHIDOS
if(!empty($_POST['usuario']) && !empty($_POST['senha'])){

//SET DA CONEXÃO LOCAL
$dsn='mysql:host=localhost;dbname=php_com_pdo';
$usuario = 'root';
$senha = '';

//INICIO DO TESTE DE CONEXÃO
try{
$conexao = new PDO($dsn, $usuario, $senha);

//SELECT E CONCATENANDO OS CAMPUS (USUARIO E SENHA) JÁ REALIZANDO A COMPARAÇÃO
NO BANCO
$query = "select * from tb_usuarios where ";
$query .= "email = '{$_POST['usuario']}'";
$query .= " AND senha = '{$_POST['senha']}'";

//ECHO DA QUERY CONCATENADA PARA TESTES
//echo $query;
```

```
//PDOStatemet  declarando a variavel e acessando o metodo query do PDO
$stmt = $conexao->query($query);
//DECLARANDO A VARIAVEL DO USUARIO UTILIZANDO A $stmt PARA ACESSAR OS METODOS
PDO E UTILIZAR AS FUNÇÕES DO PDO COMO O FETCH
$usuario = $stmt->fetch();

echo '<pre>';
print_r($usuario);
echo '</pre>';

}catch(PDOException $e){
    echo '<pre>';
    print_r($e);
    echo '</pre>';
 }
}
?>

  //HTML DO LOGIN  E SENHA
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="" method="post" action="index.php">

            <input type="text" placeholder="usuario" name="usuario">
    <br />
            <input type="password" placeholder="senha" name="senha" id="">
    <br />
            <button type="submit">Entrar</button>

    </form>
</body>
</html>
```

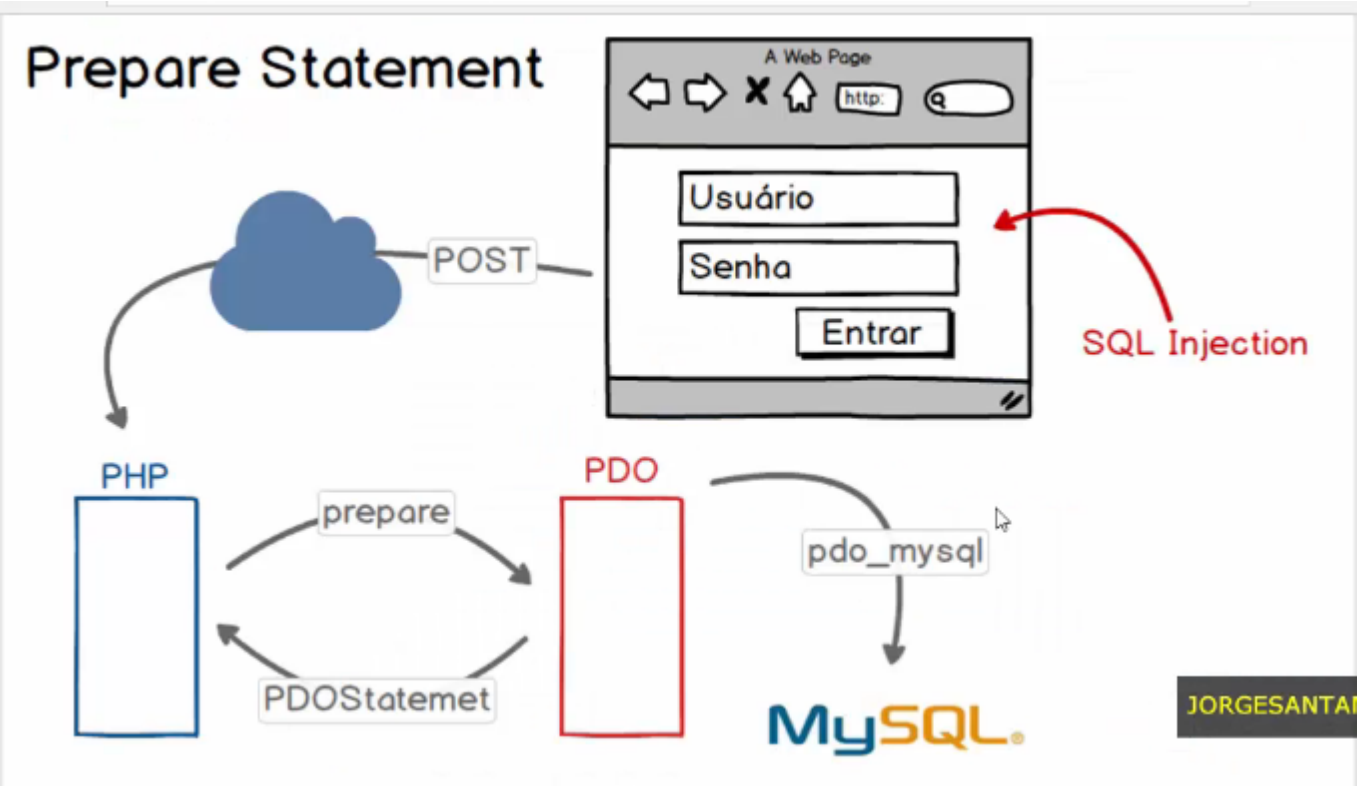
O código será utilizado no SQL injection

#SQL_Injection

Uma vulnerabilidade dos sistemas é o SQL injection sendo a inserção de comandos SQL dentro do body / request / url que fazer parte da query do backand.



É utilizado o método prepare que está nativamente no PHP para realizar a verificação e tratativa das informações que serão passadas em um query retirando qualquer injection:



1-Metodo prepare();**
O prepare retorna um PDOStmt e não executa a query inicialmente com objetivo de tratar as informações e no fim executar quando requisitado;

```
$conexao = new PDO($dsn, $usuario, $senha);
$query = "select * from tb_usuarios where ";
//:usuario e :senha são variavei utilizadas no metodo bindValue
$query .= "email = :usuario";
$query .= " AND senha = :senha";

//retorna um pdostmt o prepare não executa a query sem executar ela de outra forma
$stmt = $conexao->prepare($query);
```

2-Metodo bindValue();
Com o retorno do variável ($stmt) é utilizado o método bindValue para tratar os campos que está na variável POST sendo o campo Usuário e Senha. O bindValue vai retirar e tratar qualquer injection/comando SQL que estiver dentro do campo e por fim atribuído em uma variável utilizando " :usuario ";

```
//o campo usuario sera tratado pelo bindValue retirando qualquer injection e por fim atribuindo o valor em um bind/variavel(:usuario)
```

```
$stmt->bindValue(':usuario', $_POST['usuario']);
$stmt->bindValue(':senha', $_POST['senha']);
```

3-Metodo execute();
Utilizado no final do método prepare para executar a query.

```
//final do metodo prepare executando a query
$stmt->execute();

$usuario = $stmt->fetch();
echo '<pre>';
print_r($usuario);
echo '</pre>';
```

4-Codigo final prepare injection

```
<?php
// print_r($_POST);
//VERIFICAÇÃO SE OS CAMPOS ESTÁ PREENCHIDOS
if(!empty($_POST['usuario']) && !empty($_POST['senha'])){

//SET DA CONEXÃO LOCAL
$dsn='mysql:host=localhost;dbname=php_com_pdo';
$usuario = 'root';
$senha = '';

//INICIO DO TESTE DE CONEXÃO
try{
$conexao = new PDO($dsn, $usuario, $senha);

//SELECT E CONCATENANDO OS CAMPUS (USUARIO E SENHA) JÁ REALIZANDO A COMPARAÇÃO
NO BANCO
$query = "select * from tb_usuarios where ";
$query .= "email = :usuario";
$query .= " AND senha = :senha";

//echo $query;
//retorna um pdostmt o prepare não executa a query sem executar ela de outra
```

```
forma
$stmt = $conexao->prepare($query);

//o campo usuario sera tratado pelo bindValue retirando qualquer injection e
por fim atribuindo o valor em um bind/variavel(:usuario)
$stmt->bindValue(':usuario', $_POST['usuario']);
$stmt->bindValue(':senha', $_POST['senha']);

//final do metodo prepare executando a query
$stmt->execute();



$usuario = $stmt->fetch();
echo '<pre>';
print_r($usuario);
echo '</pre>';

}catch(PDOException $e){
    echo '<pre>';
    print_r($e);
    echo '</pre>';
 }
}
?>

//HTML DO LOGIN  E SENHA
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="" method="post" action="index.php">
    <input type="text" placeholder="usuario" name="usuario">
    <br />
    <input type="password" placeholder="senha" name="senha" id="">
    <br />
```

```
    <button type="submit">Entrar</button>
    </form>
</body>
</html>
```
