Sessões e Autenticação Simples em APIs

Prof. me. Hélio Esperidião

Session

- Chama-se variáveis de sessão aquelas que guardam valores enquanto o navegador do usuário estiver aberto.

- Ao encerrar o navegador estas variáveis são perdidas.
- O interessante destas sessões é que elas podem ser acessadas por qualquer página de sua aplicação PHP.

Como uma sessão funciona?

- Em um site ou sistema web a sessão é importante quando se quer ter um controle de usuário.

- É guardar informações entre requisições, criar áreas restritas, montar um carrinho de compras, etc.

Exemplo

Banco de dados.

```
CREATE DATABASE aulaPawLogin;
USE aulaPawLogin;
CREATE TABLE usuario(
idUsuario INT AUTO_INCREMENT PRIMARY KEY,
nome VARCHAR(128),
email VARCHAR(128),
senha CHAR (32)
)
```

Select para validar usuário

```
1 select idUsuario, nome,email,senha, COUNT(*) AS qtd  from usuario
2 WHERE email = 'helioesperidiao@gmail.com' AND senha ='123456'
```

usuario (1r × 5c)

idUsuario	nome	email	senha	qtd
7	helio	helioesperidiao@gmail.com	123456	1

```
1 select idUsuario, nome,email,senha, COUNT(*) AS qtd  from usuario
2 WHERE email = 'helioesperidiao@gmail.com' AND senha ='1234567'
```

usuario (1r × 5c)

idUsuario	nome	email	senha	qtd
(NULL)	(NULL)	(NULL)	(NULL)	0

Banco.php

```
class Banco{
    private $host     = "127.0.0.1";
    private $usuario  = "root";
    private $senha    = "";
    private $banco    = "aulaPawLogin";
    private $porta     = "3306";
    private $con=null;
    private function conectar(){
        $this->con = new mysqli( $this->host, $this->usuario,$this->senha,$this->banco, $this->porta);
        if ( $this->con->connect_error) {
            die("Falha ao conectar: " .  $this->con->connect_error);
        }
    }
    public function getConexao(){
        if( $this->con==null){
            $this->conectar();
        }
        return $this->con;
    }
}
```

Atributos da classe de conexão com o banco

Executa quando o método getConexao() é chamado
e o atributo $con é igual a null
Método privado é executado somente dentro da instância.

Cria uma conexão com o banco.
A conexão é armazenada no atributo privado $con

Método que retorna a conexão uma conexão com o banco.
Sempre é chamado quando se deseja executar código sql

Usuario.php

```
include "Banco.php";
class Usuario implements JsonSerializable
{
    private $idUsuario;
    private $nome;
    private $email;
    private $senha;
    private $banco;
```

jsonSerialize()

```
/*
    Quando for chamado o json_decode($objUsuario)
    o método  jsonSerialize() será chamado
    Um json no formato definido em  jsonSerialize()
    será criado
  */
  public function jsonSerialize()
  {
     $json = array();
     $json['idUsuario'] = $this->getIdUsuario();
     $json['nome'] = $this->getNome();
     $json['email'] = $this->getEmail();
     return $json;
  }
```

create()

```
/*
    método chamado pelo arquivo index.php
    quando é recebido um post para cadastro
    de novo usuário.
 */
public function create()    {
    //Instancia a classe banco
    $this->banco = new Banco();
    //cria um hash da senha usando o algoritmo md5
    $this->senha = md5($this->senha);
    //prepara a instrução de insert no banco
    $stmt = $this->banco->getConexao()->prepare("insert into usuario (nome, email, senha) values (?,?,?)");
    //substitui os "s" pelos interrogações e valore de variáveis.
    $stmt->bind_param("sss", $this->nome, $this->email, $this->senha);
    //execura a instrução no banco de dados.
    $resposta = $stmt->execute();
    //retorna o id do registro que foi inserido no banco.
    $idCadastrado = $this->banco->getConexao()->insert_id;
    //passa para a instancia da classe o id que foi cadastrado
    $this->setIdUsuario($idCadastrado);
    return $resposta; //passa verdadeiro ou falso, verdadeiro se a instrução foi executada com sucesso.
}
```

usuarioExiste()

```
public function usuarioExiste(){
    $this->banco = new Banco();  //faz uma instância de banco
    $sql = "select count(*) as qtd from usuario where email = ?";
    $stmt = $this->banco->getConexao()->prepare($sql);   //prepara a instrução de select no banco
     $stmt->bind_param("s", $this->email);  //vincula os parâmetros substituindo os "?" pelos valores correspondentes.
    $stmt->execute();  //executa a  instrução sql no sgbd
    $resultado = $stmt->get_result(); //recupera os dados referentes a execução do sql
    while ($linha = $resultado->fetch_object()) { //estrutura de repetição que passa por todos os dados
       //verifica se qtd é igual a 1. igual a 1 significa que existe 1 usuário com o email fornecido.
        if ($linha->qtd == 1) {
            return true;
        }
    }
    return false;
}
```

delete()

```
/*
    método chamado pelo arquivo index.php
    quando é recebido um Delete para exclusão
    de usuário.
 */
public function delete()
{
    $this->banco = new Banco();  //Instancia a classe banco
    //prepara a instrução de delete no banco
    $stmt = $this->banco->getConexao()->prepare("delete from usuario where idUsuario = ?");
    //substitui o ? pelo "i"(inteiro) que corresponde ao idUsuario
    $stmt->bind_param("i", $this->idUsuario);
    //retorna verdadeiro se a instrução foi executa com sucesso no sgbd
    return $stmt->execute(); //executa a instrução no sgbd
}
```

update()

```
/*
    método chamado pelo arquivo index.php
    quando é recebido um PUT para atualização
    de usuário.
 */
public function update(){
    $this->banco = new Banco(); //Instancia a classe banco
    $this->senha = md5($this->senha);  //cria um hash da senha usando o algoritmo md5
    //prepara a instrução de update no banco
    $stmt = $this->banco->getConexao()->prepare("update usuario
                        set nome=?,
                        email=?,
                        senha=?
                        where idusuario = ?");
    //substitui os "s,i" pelos interrogações e valores dos atributos.
    $stmt->bind_param("sssi", $this->nome, $this->email, $this->senha, $this->idUsuario);
    return $stmt->execute();  //executa a instrução no sgbd
}
```

read()

```
/*  método chamado pelo arquivo index.php quando é recebido um GET com id do Usuario para retornar dados do usuário com id corespondente. */
  public function read(){
    $this->banco = new Banco(); //Instancia a classe banco
    //prepara a instrução de select no banco
    $stmt = $this->banco->getConexao()->prepare("select *  from usuario where idUsuario=?");
    //substitui os "i" pelo interrogação e valor da variável.
    $stmt->bind_param("i", $this->idUsuario);
    $stmt->execute(); //executa a instrução no sgbd
    $resultado = $stmt->get_result(); //recupera os dados referentes a execução da instrução.
    //cria um vetor de objetos com os do usuário de um id especifico.
    $usuario= array();
    while ($linha = $resultado->fetch_object()) {
      //é utilizado apenas o índice zero pois  só deve existir um único usuário com o id especifico.
      $usuario[0] = new Usuario();
      $usuario[0]->setIdUsuario($linha->idUsuario);
      $usuario[0]->setNome($linha->nome);
      $usuario[0]->setEmail($linha->email);
    }
    return   $usuario; //retorna um vetor com os dados do usuário.
  }
```

readAll()

```
// método chamado pelo arquivo index.php quando é recebido um GET sem o id do Usuario para retornar dados de todos os usuario.
public function readAll(){
    $this->banco = new Banco(); //Instancia a classe banco
    //prepara a instrução de select no banco
    $stmt = $this->banco->getConexao()->prepare("select *  from usuario ");
    $stmt->execute();  //executa a instrução no sgbd
    $resultado = $stmt->get_result();  //recupera os dados referentes a execução da instrução.
    $usuario = array(); $i = 0;
    //cria um vetor com os dados de todos os usuários
    while ($linha = $resultado->fetch_object()) {
        $usuario[$i] = new Usuario(); //instância um novo usuário em cada posição do vetor
        //recupera os dados que vieram do sgbd  e faz o "set" dos dados
        $usuario[$i]->setIdUsuario($linha->idUsuario);
        $usuario[$i]->setNome($linha->nome);
        $usuario[$i]->setEmail($linha->email);
        $i++;
    }
    return   $usuario; //retorna um vetor de usuários
}
```

logout()

```
/*
    método chamado pelo arquivo index.php
    quando é recebido um GET na rota /logout
    para destrurir a sessão.
*/
public function logout()
{
    session_unset();
    session_destroy();
}
```

estaLogado()

```
//método que verifica se o usuário tem sessão
// e foi logado previamente
public function estaLogado()
{
   //verifica se existe a variável $_SESSION["logado"]
   if (isset($_SESSION["logado"])) {
      //verifica se o valor da variável $_SESSION["logado"] é true
      if ($_SESSION["logado"] == true) {
         return true;
      }
   }
   return false;
}
```

verificarUsuarioSenha()

```
//método que verifica se o usuário e senha estão corretos
  public function verificarUsuarioSenha(){
     $this->banco = new Banco(); //faz uma instância de banco
     $this->senha = md5($this->senha); //cria um hash utilizando o algoritmo md5 para a senha
     $sql = "select count(*) as qtd ,nome,idUsuario  from usuario where email = ? and senha =?";
     $stmt = $this->banco->getConexao()->prepare($sql);
     $stmt->bind_param("ss", $this->email, $this->senha);
     $stmt->execute();  //executa a  instrução sql no sgbd
     $resultado = $stmt->get_result();   //recupera os dados referentes a execução do sql
     //estrutura de repetição que passa por todos os dados
     while ($linha = $resultado->fetch_object()) {
        //verifica se qtd é igual a 1, significa que existe 1 usuario om o email e senha fornecido.
        if ($linha->qtd == 1) {
           $this->setIdUsuario($linha->idUsuario);
           $this->setNome($linha->nome);
           //cria as variáveis de sessão do usuário logado.
           $_SESSION["logado"] = true;
           $_SESSION["idUsuario"] =  $this->getIdUsuario();
           $_SESSION["email"] =  $this->getEmail();
           $_SESSION["nome"] = $this->getNome();
           return true;
        }
     }
     return false;
  }
```

Index.php

```php
<?php
//inicializa uma sessao
session_start();
//https://github.com/bramus/router
require_once "modelo/Router.php";
require_once "modelo/Usuario.php";

//Cria uma instância da classe Router
$router  = new Router();
```

Index.php

```
// trata a rota: POST: /usuario, a rota cadastra um cliente.
$router->post('/usuario', function () {
$jsonRecebido = file_get_contents('php://input'); //recupera os dados enviados no corpo da requisição.
  $obj = json_decode($jsonRecebido);   //converte os dados em um objeto json.
  $u1 = new Usuario();   //cria uma instância de Usuario
 $u1->setNome($obj->nome); //faz o "set" dos dados dentro da classe.
  $u1->setEmail($obj->email);
  $u1->setSenha($obj->senha);
  //verifica se existe algum usuário com o mesmo email
  if ($u1->usuarioExiste() == false) {   //executa o método create da classe Usuário e prepara o array resposta
que será convertido em json
    $resposta['status'] = $u1->create();
    $resposta['msg'] = 'cadastrado com sucesso';
    $resposta['dados'] = $u1;
  } else {
    $resposta['status'] = false;
    $resposta['msg'] = 'já existe um usuário com o e-mail fornecido';
    $resposta['dados'] = $u1;
  }
  echo json_encode($resposta); // converte o array em um json
});
```

Index.php

```
// trata a rota: GET /usuario, retorna todos os usuários
$router->get('/usuario', function () {
  $u1 = new Usuario();
  $resposta['status'] = true;
  $resposta['dados'] =  $u1->readAll();
  echo json_encode($resposta);
});
```

Index.php

```
//trata a rota: GET /usuario/idUsuario, retorna usuário com id especifico
$router->get('/usuario/(\d+)', function ($v1) {
  $u1 = new Usuario();
  $u1->setIdUsuario($v1);
  $resposta['status'] = true;
  $resposta['dados'] =  $u1->read();
  echo json_encode($resposta);
});
```

Index.php

```
//trata a rota: PUT /usuario/idUsuario
$router->put('/usuario/(\d+)', function ($v1) {
  $u1 = new Usuario();
  if ($u1->estaLogado() == true) {
    $jsonRecebido = file_get_contents('php://input');
    $obj = json_decode($jsonRecebido);
    $u1->setIdUsuario($v1);
    $u1->setNome($obj->nome);
    $u1->setEmail($obj->email);
    $u1->setSenha($obj->senha);
    $resposta['status'] = $u1->update();
    $resposta['msg'] = "Atualizado com sucesso";
    $resposta['dados'] = $u1;
    echo json_encode($resposta);
  } else {
    $resposta['status'] = false;
    $resposta['msg'] = 'Usuário não logado';
    echo json_encode($resposta);
  }
});
```

Index.php

```
////trata a rota: DELETE /usuario/idUsuario
$router->delete('/usuario/(\d+)', function ($v1) {
  $u1 = new Usuario();
  if ($u1->estaLogado() == true) {
    $u1->setIdUsuario($v1);
    $resposta['status'] = $u1->delete();
    $resposta['msg'] = "excluído com sucesso";
    echo json_encode($resposta);
  } else {
    $resposta['status'] = false;
    $resposta['msg'] = 'Usuário não logado';
    echo json_encode($resposta);
  }
});
```

Index.php

```
//verifica se usuário e senha estão corretos
//cria uma sessão para o usuário
$router->post('/login', function () {
  $jsonRecebido = file_get_contents('php://input');
  $obj = json_decode($jsonRecebido);
  $u1 = new Usuario();
  $u1->setEmail($obj->email);
  $u1->setSenha($obj->senha);
  $resposta = array();
  if ($u1->verificarUsuarioSenha() == true) {
    $resposta['status'] = 'true';
    $resposta['msg'] = "Login efetuado com sucesso";
  } else {
    $u1->logout();
    $resposta['status'] = false;
    $resposta['msg'] = "Usuario ou senha invalidos";
  }
  echo json_encode($resposta);
});
```

Index.php

```
//trata a rota /logout
//destroi a sessão
$router->get('/logout', function () {

  $u1 = new Usuario();
  $u1->logout();

  $resposta = array();
  $resposta['status'] = true;
  $resposta['msg'] = "sessão encerrada";
  echo json_encode($resposta);
});
$router->run();
```