Hoje vou mostrar como utilizar esse conhecimento para desenvolver um sistema de login em PHP.

Sistemas de login são uma das principais aplicações das Sessões, por isso optei por falar sobre esse tema.

**Como Será Nosso Sistema de Login**

Faremos um sistema de login simples, porém bem estruturado.

Teremos uma tela inicial, um formulário de login e uma tela protegida, com o Painel do Usuário, acessível apenas por quem está logado.

Teremos a seguinte estrutura de arquivos.

* check.php: protege scripts, impedindo acesso por visitantes não logados
* form-login.php: formulário de login
* functions.php: arquivo com funções que usaremos em nosso sistema de login
* index.php: página inicial
* init.php: arquivo de inicialização, onde vamos definir constantes e outras configurações gerais
* login.php: script que verifica os dados de login fornecidos no formulário
* logout.php: script para fazer *logout*
* panel.php: painel do usuário

Parece muita cosia?

Calma…

É coisa simples.

Vamos ver parte por parte, uma de cada vez.

Criando o Banco de Dados

Antes de tudo, vamos criar nosso banco de dados.

Vou usar o MySQL, por ser o mais utilizado.

Coisa simples, só uma tabela com ID, nome, email e senha.

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE users(      id INT UNSIGNED NOT NULL AUTO\_INCREMENT,      name VARCHAR(50),      email VARCHAR(80) NOT NULL,      password VARCHAR(40) NOT NULL,      PRIMARY KEY(id)  ) COLLATE=utf8\_unicode\_ci; |

Vamos inserir alguns registros para nossos testes.

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO users(name, email, password) VALUES  ('Usuário 1', 'user1@user.com', '2d29b962490320f821db80cddf6ed4b6e4ac7498'),  ('Usuário 2', 'user2@user.com', '2d29b962490320f821db80cddf6ed4b6e4ac7498'),  ('Usuário 3', 'user3@user.com', '2d29b962490320f821db80cddf6ed4b6e4ac7498'); |

Essa senha é um hash SHA-1 de um hash MD5 da string “123mudar”, que defini como senha padrão.

Se quiser verificar, basta executar esta chamada PHP:

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1 | echo sha1(md5('123mudar')); |

É **FUNDAMENTAL** não salvar senhas em texto puro. Use sempre algum tipo de *hash*.

**Arquivo de Funções**

Antes de tudo, vamos criar nossas funções no arquivo functions.php.

Precisamos de funções para conexão com o banco de dados, para gerar o hash da senha e para verificar se o usuário está logado.

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | <?php    /\*\*   \* Conecta com o MySQL usando PDO   \*/  function db\_connect()  {      $PDO = new PDO('mysql:host=' . DB\_HOST . ';dbname=' . DB\_NAME . ';charset=utf8', DB\_USER, DB\_PASS);        return $PDO;  }      /\*\*   \* Cria o hash da senha, usando MD5 e SHA-1   \*/  function make\_hash($str)  {      return sha1(md5($str));  }      /\*\*   \* Verifica se o usuário está logado   \*/  function isLoggedIn()  {      if (!isset($\_SESSION['logged\_in']) || $\_SESSION['logged\_in'] !== true)      {          return false;      }        return true;  } |

A função db\_connect abre a conexão com o MySQL, usando a biblioteca PDO. Se você não conhece PDO, sugiro ler [este meu artigo](http://rberaldo.com.br/pdo-mysql/), onde explico, em detalhes, como usá-la com MySQL. As constantes DB\_HOST, DB\_NAME, DB\_USER e DB\_PASS nós vamos criar no arquivo init.php no próximo passo.

A função make\_hash é parecida com o trecho que eu mostrei um pouco mais cedo. Essa função apenas retorna o *hash* de uma string. Usaremos essa função para gerar o hash da senha digitada pelo usuário.

A função isLoggedIn verifica se o usuário está logado. Para isso, ela usa o valor de logged\_in no array $\_SESSION. Nós vamos criar esse índice com o valor true sempre que o visitante efetuar login.

**Arquivo de Inicialização**

Aqui vamos criar o arquivo init.php, definindo algumas configurações gerais do nosso projeto.

Isso inclui as credenciais de acesso ao banco de dados.

Mas você pode colocar neste arquivo qualquer outra configuração do PHP, como definir *timezone*, habilitar e desabilitar exibições de erro e por aí vai.

O arquivo de inicialização deve ser chamado em todos os nossos scripts. Assim teremos um ambiente consistente e organizado, sempre com todas as mesmas configurações.

Se quiser ir um pouco mais a fundo nesse assunto, recomendo ler [este meu artigo](http://rberaldo.com.br/bootstrapping-php-arquivo-inicializacao/), onde falo sobre Bootstrapping. Mostro algumas técnicas um pouco mais avançadas sobre esse tema.

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?php    // constantes com as credenciais de acesso ao banco MySQL  define('DB\_HOST', 'localhost');  define('DB\_USER', 'root');  define('DB\_PASS', '');  define('DB\_NAME', 'ultimatephp\_artigos');    // habilita todas as exibições de erros  ini\_set('display\_errors', true);  error\_reporting(E\_ALL);    // inclui o arquivo de funçõees  require\_once 'functions.php'; |

Note que incluímos o funcions.php. Dessa forma, sempre que incluirmos o init.php nos nossos scripts, também teremos acesso às nossas funções personalizadas.

**Página Inicial do Sistema de Login**

Nossa página inicial (arquivo index.php) terá apenas uma mensagem de boas vindas e alguns links.

Se o usuário não estiver logado, mostraremos o link de login.

Caso ele já esteja logado, mostraremos seu nome e os links do Painel e de Sair.

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <?php  session\_start();    require 'init.php';  ?>  <!doctype html>  <html>      <head>          <meta charset="utf-8">            <title>Sistema de Login ULTIMATE PHP</title>      </head>        <body>            <h1>Sistema de Login ULTIMATE PHP</h1>            <?php if (isLoggedIn()): ?>              <p>Olá, <?php echo $\_SESSION['user\_name']; ?>. <a href="panel.php">Painel</a> | <a href="logout.php">Sair</a></p>          <?php else: ?>              <p>Olá, visitante. <a href="form-login.php">Login</a></p>          <?php endif; ?>        </body>  </html> |

**Formulário de Login**

Vamos ao formulário de login…

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | <!doctype html>  <html>      <head>          <meta charset="utf-8">            <title>Login | Sistema de Login ULTIMATE PHP</title>      </head>        <body>            <h1> Login</h1>            <form action="login.php" method="post">              <label for="email">Email: </label>              <br>              <input type="text" name="email" id="email">                <br><br>                <label for="password">Senha: </label>              <br>              <input type="password" name="password" id="password">                <br><br>                <input type="submit" value="Entrar">          </form>        </body>  </html> |

Esse é um simples HTML, sem segredos.

Apenas um formulário que leva ao nosso login.php.

**Efetuando o Login**

Agora vamos para a parte que realmente interessa mais: o *login* propriamente dito.

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | <?php    // inclui o arquivo de inicialização  require 'init.php';    // resgata variáveis do formulário  $email = isset($\_POST['email']) ? $\_POST['email'] : '';  $password = isset($\_POST['password']) ? $\_POST['password'] : '';    if (empty($email) || empty($password))  {      echo "Informe email e senha";      exit;  }    // cria o hash da senha  $passwordHash = make\_hash($password);    $PDO = db\_connect();    $sql = "SELECT id, name FROM users WHERE email = :email AND password = :password";  $stmt = $PDO->prepare($sql);    $stmt->bindParam(':email', $email);  $stmt->bindParam(':password', $passwordHash);    $stmt->execute();    $users = $stmt->fetchAll(PDO::FETCH\_ASSOC);    if (count($users) <= 0)  {      echo "Email ou senha incorretos";      exit;  }    // pega o primeiro usuário  $user = $users[0];    session\_start();  $\_SESSION['logged\_in'] = true;  $\_SESSION['user\_id'] = $user['id'];  $\_SESSION['user\_name'] = $user['name'];    header('Location: index.php'); |

Vejamos alguns pontos importantes.

Na linha 17, eu uso a nossa função make\_hash, para gerar o *hash* da senha informada pelo visitante.

Em seguida, abrimos a conexão com o MySQL usando nossa função db\_connect e efetuamos um SELECT, buscando ID e nome do usuário, filtrando pelo email e pelo hash da senha. Selecionei apenas ID e nome, pois os outros dados (email e senha) eu já tenho. É importante tomar cuidado para selecionar apenas os dados necessários. Isso ajuda no desempenho da aplicação.

Nas linhas 40 a 43, após verificação das credenciais, a sessão é iniciada e criamos as variáveis de sessão. Note que aqui eu crio o índice logged\_in, que é o mesmo que usamos na função isLoggedIn.

Em seguida, redireciono o usuário para a página inicial.

A partir desse momento, você já pode testar o login. Você verá que, efetuar login com um dos emails e com a senha “123mudar”, o login será bem sucedido e a mensagem de boas-vindas mostrará o nome do usuário logado.

**Efetuando *Logout***

Tão importante quanto o login é o *logout* (opção “Sair”).

Vamos criar o script logout.php, com este conteúdo:

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <?php    // inicia a sessão  session\_start();    // muda o valor de logged\_in para false  $\_SESSION['logged\_in'] = false;    // finaliza a sessão  session\_destroy();    // retorna para a index.php  header('Location: index.php'); |

Na linha 4 a sessão é iniciada. Como eu disse no [artigo sobre sessões](http://blog.ultimatephp.com.br/como-usar-sessoes-php/), a sessão deve ser sempre iniciada, mesmo quando queremos apenas finalizá-la.

Na linha 7, definimos o valor de logged\_in como false. Essa linha é opcional, já que em seguida vamos destruir a sessão. Ou seja, todos os índices de $\_SESSION serão apagados. Mais adianta vou explicar por que prefiro manter essa linha.

Em seguida destruimos a sessão e redirecionamos o usuário para a página inicial.

**Painel do Usuário**

A última página que faltava…

Na verdade, não colocarei nenhuma funcionalidade neste painel. Minha intenção é apenas mostrar um detalhes **importantíssimo**.

Crie o arquivo panel.php com este conteúdo:

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <?php  session\_start();    require\_once 'init.php';  ?>  <!doctype html>  <html>      <head>          <meta charset="utf-8">            <title>Painel | Sistema de Login ULTIMATE PHP</title>      </head>        <body>            <h1>Painel do Usuário</h1>            <p>Bem-vindo ao seu painel, <?php echo $\_SESSION['user\_name']; ?> | <a href="logout.php">Sair</a></p>      </body>  </html> |

Agora você pode efetuar login e acessar o painel.

Porém…

Temos uma **grave** falha de Segurança.

Experimente fazer logout e acessar diretamente o arquivo panel.php.

Viu o problema?

É possível acessar esse script mesmo sem estar logado.

**Restringindo Acesso**

Para arrumar nossa falha de Segurança, vamos criar o arquivo check.php.

Esse arquivo fará algo bem simples: verificar se o usuário está logado. Se não estiver, redireciona para o formulário de login.

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php    require\_once 'init.php';    if (!isLoggedIn())  {      header('Location: form-login.php');  } |

Depois basta incluir esse arquivo em nosso panel.php, ficando assim:

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <?php  session\_start();    require\_once 'init.php';    require 'check.php';  ?>  <!doctype html>  <html>      <head>          <meta charset="utf-8">            <title>Painel | Sistema de Login ULTIMATE PHP</title>      </head>        <body>            <h1>Painel do Usuário</h1>            <p>Bem-vindo ao seu painel, <?php echo $\_SESSION['user\_name']; ?> | <a href="logout.php">Sair</a></p>      </body>  </html> |

Pronto.

Agora, ao tentar acessar o painel sem estar logado, o sistema vai redirecionar o visitante para o formulário de login.

**Dicas e Orientações**

Uma coisa chata é ficar chamando session\_start em todos os scripts.

Já que temos um arquivo de inicialização, podemos fazer essa chamada apenas lá, uma única vez. Assim podemos tirar todas as demais chamadas para session\_start.

Nesse caso, é bom sempre tomar cuidado na verificação de login, já que a sessão estará sempre aberta.

Foi por isso que criei o índice logged\_ind.

Usando essa abordagem, poderíamos até tirar o session\_destroy.

Aliás, pra organizar tudo, eu recomendo fazer o seguinte:

Crie um array dentro de $\_SESSION, que vai armazenar todas as variáveis relacionadas ao login.

Assim você pode remover esse índice inteiro no logout, mantendo a sessão aberta, sem influenciar outros dados que possam estar na sessão.

Por exemplo, imagine que você, além do login, também salva na sessão alguns dados de formulários, idioma do site etc.

Nesse caso, você poderia fazer algo assim:

[?](http://blog.ultimatephp.com.br/sistema-de-login-php/)

|  |  |
| --- | --- |
| 1  2  3  4 | $\_SESSION['lang'] = 'pt\_br';    $\_SESSION['login']['logged\_in'] = true;  $\_SESSION['login']['user\_name'] = 'Usuário 1'; |

Note que todas as variáveis que usamos antes agora estão dentro do array $\_SESSION['login'].

Ou seja, no logout, bastaria apagar esse array, usando unset($\_SESSION['login']). Isso faz o logout, sem destruir a sessão e sem perder a configuração de idioma, salva em $\_SESSION['lang'].