Programação II

Avaliação a distância 2 – GABARITO (06/06/2007)

1. Sabemos que sessões precisam ter um limite de tempo de inatividade após o qual a sessão deve ser fechada automaticamente. Por quê isso é necessário?.

   Resp.:
   Porque as variáveis de sessão são guardadas no servidor tomando recursos. Como não há garantia de que o cliente irá terminar a sessão explicitamente, o servidor libera esses recursos e fecha a sessão após um tempo pré-determinado de inatividade.

2. Suponha que você está implementando uma aplicação web que solicita o preenchimento de um formulário com os campos `nome`, `email`, `idade` e `CEP`. Escreva em PHP o código relevante para apresentação do formulário e para validação do mesmo. Caso algum item não passe na validação, o formulário deve ser reapresentado com os nomes dos campos incorretos marcados com um asterisco. É importante também que, nesse caso, os valores iniciais de todos os campos sejam aqueles originalmente informados pelo usuário. As regras para validação são:

   `Nome` deve conter ao menos duas palavras com mais de dois caracteres, sendo que todos os caracteres devem ser alfabéticos (não é necessário testar caracteres acentuados).

   `Email` deve conter exatamente um caractere '@', sendo que o código do usuário deve ser alfanumérico ou conter os caracteres '-', '.' ou '_'. O domínio (parte após o '@') deve conter apenas caracteres alfanuméricos, '-' ou '.', sendo que ao menos um ponto é obrigatório e pontos não podem aparecer nem no início nem no fim do domínio.

   `Idade` deve ser numérica e superior a 18 (anos).

   `CEP` deve ser constituído de exatamente 8 caracteres numéricos.

   Resp.:

```
<?php

/// seleciona a opção a processar
switch($_POST['acao']){
   case 'enviar':
      validaForm($_POST);
      break;
   default:
      exibeForm();
      break;
}

/// válida os dados de um form enviados pelo método POST
/// @param post Array contendo os dados do form
function validaForm($post){
   $error = array();
   // requisito nomes com mais de duas palabras
   // e mais de duas letras em cada palavra
```

```
   // (verificar supporte POSIX na sua versão de php para evitar uso dos
   // carateres especiais ãõáéíóúêôâÃÕÁÉÍÓÚÊÔ)
   if (trim( $post['nome'] ) == '' ||
      !ereg(" *[a-zA-Zãõáéíóúêôâ ÃÕÁÉÍÓÚÊÔ\.]{2,} +[a-zA-
Zãõáéíóúêôâ ÃÕÁÉÍÓÚÊÔ\.]{2,}", $post['nome']) ){
      $error['nome'] = "*";
   }

   if (trim( $post['email'] ) == '' ||
      !eregi("^[a-zA-Z0-9]+[_a-zA-Z0-9-]*(\.[_a-z0-9-]+)*@[a-zA-Z0-9]+(-[a-zA-Z0-
9]+)*(\.[a-zA-Z0-9-]+)*(\.[a-z]{2,4})$", $post['email']) ) {
      $error['email'] = "*";
   }

   if (trim( $post['idade'] ) == '' || !ereg("^[0-9]+$", $post['idade']) ||
(int)($post['idade']) < 18 ){
      $error['idade'] = "*";
   }

   if ( $post['cep'] == '' || !ereg("^[0-9]{8}$", $post['cep'])){
      $error['cep'] = "*";
   }



   // se houver erro, re-exibe o form indicando os campos onde houve erro
   // caso contrario, exibe uma tabela mostrando os dados indicando sucesso
   if (sizeof($error))
      exibeForm($post, $error);
   else
      exibeResultado($post);
}

/// Exibe um form
/// @param vals Array contendo os valores dos campos no form
/// @param errors Array contendo um indicador no caso
/// de haver erro no conteudo do campo
function exibeForm($vals = array(), $errors = array() ){
?>
   <form action='questao2.php' method='post' name='Form'>
   <table>
   <tbody>
   <tr>
      <td colspan='3'>Ficha para preenchimento.</td>
   </tr>

   <tr>
      <td width='30%'>
         Nome:
      </td>
      <td>
         <input name='nome' size='40' value='<?php echo $vals['nome']; ?>'
```

```
                maxlength='100' type='text'> <?php echo $errors['nome']; ?>
         </td>
      </tr>
      <tr>
         <td>
            E-mail:
         </td>
         <td>
            <input name='email' size='40' value='<?php echo $vals['email']; ?>'
                   maxlength='100' type='text'> <?php echo $errors['email']; ?>
         </td>
      </tr>
      <tr>
         <td>
            Idade
         </td>
         <td>
            <input name='idade' size='40' value='<?php echo $vals['idade']; ?>'
                   maxlength='25' type='text'> <?php echo $errors['idade']; ?>
         </td>
      </tr>
      <tr>
         <td>
            CEP
         </td>
         <td>
            <input name='cep' size='40' value='<?php echo $vals['cep']; ?>'
                   maxlength='8' type='text'> <?php echo $errors['cep']; ?>
         </td>
      </tr>
      </tbody>
      </table>
      <input name="acao" value="enviar" type="hidden">
      <input value="Enviar" class="button" type="submit">
      </form>
<?php
}

/// Exibe uma tabela de dados
/// @param vals Array contendo os dados a serem exibidos
function exibeResultado($vals){
?>
      <table>
      <tbody>
      <tr>
         <td colspan='3'>Dados recebidos.</td>
      </tr>

      <tr>
         <td width='30%'>
```

```
            Nome:
         </td>
         <td>
            <?php echo $vals['nome']; ?>
         </td>
      </tr>
      <tr>
         <td>
            E-mail:
         </td>
         <td>
            <?php echo $vals['email']; ?>
         </td>
      </tr>
      <tr>
         <td>
            Idade
         </td>
         <td>
            <?php echo $vals['idade']; ?>
         </td>
      </tr>
      <tr>
         <td>
            CEP
         </td>
         <td>
            <?php echo $vals['cep']; ?>
         </td>
      </tr>
      </tbody>
      </table>
<?php
}
?>
```

3. Um problema freqüente em aplicações web é a codificação de caracteres especiais e acentuados, dificultando a visualização da aplicação web. Como se pode garantir que uma aplicação web mostrará sempre os caracteres especiais corretamente?

   Resp.:

   Freqüentemente o servidor onde a aplicação está hospedada tem configurado um charset diferente daquele usado para editar a aplicação. Por exemplo, o servidor pode ter configurado UTF-8 como o charset default e a aplicação ter sido escrita usando ISO-8859-1 (ou vice-versa). Como para alterar o charset do servidor requer privilégio de administrador, é mais conveniente explicitar o charset usado na aplicação usando *headers*. Por exemplo, uma aplicação escrita usando ISO-8859-1 deveria ser iniciada por algo como:

```
<?php
...
header("Content-Type: text/html; charset=ISO-8859-1");
...
?>
```

4. Considere uma aplicação de gerência de empréstimos de livros em uma biblioteca. Suponha que num banco de dados chamado `bib` haja uma tabela `livro` com os seguintes campos: `Titulo` (varchar (40)) e `NumExemplares` (int). Escreva as seguintes funções em PHP:

   `function consulta_titulo ($titulo)`: retorna o número de exemplares do livro com `Titulo = $titulo`. Deve retornar -1 caso o livro não exista.

   `function empresta ($titulo)`: decrementa o numero de exemplares do livro correspondente e retorna `true` se a atualização foi feita com sucesso, isto é, se a operação pôde ser feita resultando em um número de exemplares não negativo. Caso contrário, o registro do livro não é alterado e a função retorna `false`. Observe que é necessário usar "locks" para implementar esta função corretamente.

   Resp.:

```
<?php
...

/// consulta para saber se existe exemplares de um livro
/// @param titulo O titulo do livro a ser consultado
function consulta_titulo($titulo){
   $query = "select titulo, exemplares from livro where titulo = '$titulo'";
   $result = mysql_query($query);
   if ( mysql_num_rows($result) ){
      $row = mysql_fetch_assoc($result);
      if ($row['exemplares'] != 0)
         return $row['exemplares'];
   }
   return -1;
}

/// consulta para empréstimo de um exemplar de livro
/// @param titulo O titulo do livro a ser emprestado
function empresta($titulo){
   global $usuario; // usuario logado

   $exemplares = consulta_titulo($titulo);
   if ($exemplares != -1){
      $query = "LOCK TABLES livro WRITE, emprestimos WRITE";
      mysql_query($query);

      $exemplares = $exemplares - 1; // deminuir em 1 o número de exemplares
      $query = "UPDATE livro SET exemplares = $exemplares ".
               "WHERE titulo = $titulo";
      mysql_query($query);
```

```
      // Adicionalmente deve-se atualizar a tabela de emprestimos.
      // Ela deve registrar o nome e identificador do usuário
      // que se emprestou o livro
      $query = "INSERT INTO emprestimos (id_usuario, nome, titulo) ".
               "values ($usuario['id'], $usuario['nome'], $titulo)";
      mysql_query($query);

      $query = "UNLOCK TABLES";
      mysql_query($query);
      return true;
   }

   return false;
}

...

?>
```

5. Descreva sucintamente as vantagens e desvantagens dos seguintes métodos de autenticação:

   Autenticação http usando arquivo .htaccess

   Resp.:
   É simples de usar e pode ser usado para proteger todas as aplicações residentes em um diretório sem necessidade de se modificar as aplicações em si. O fato das informações estarem em um arquivo representa um risco de segurança. Como é preciso usar ferramentas externas para criar o arquivo .htaccess, o processo não pode ser conduzido adequadamente pela aplicação. Ademais, a aplicação não é possível atribui privilégios diferentes a categorias de usuário (granularidade).

   Autenticação http em php com senhas guardadas em banco de dados

   Resp.:
   Permite um maior controle sobre a granularidade do processo de autenticação e permite um controle estrito sobre o cadastro dos privilégios de cada usuário. Entretanto, o fato de a autenticação http armazenar as credenciais no navegador do cliente representa um risco de segurança. Outra desvantagem é que o diálogo de credenciamento tem formato fixo e, portanto, pouco flexível.

   Autenticação com sessões

   Resp.:
   Elimina as principais desvantagens da autenticação com http ao armazenar as credenciais no servidor e colocar o diálogo de autenticação a cargo da aplicação. Também permite diversos níveis de autenticação que dependem apenas da aplicação. Entretanto, o tráfego entre o navegador e o servidor pode ser capturado por terceiros e, como as informações sigilosas não são criptografadas, credenciais podem ser roubadas.

Autenticação com sessões usando SSL

Resp.:
Reduz o risco de captura de informações sigilosas através da encriptação de todo o tráfego entre cliente e servidor.