PHP: Formulários e upload de múltiplos arquivos

Por Alfred Reinold Baudisch – 28 de fevereiro de 2006.

Veja como montar um formulário para upload de múltiplos arquivos e como processá-lo com o PHP, usando o exemplo de uma galeria de fotos em que é enviada uma quantidade variável de fotos.

Publiquei três artigos ensinando como processar dados de formulário com o PHP, sendo dois deles sobre o upload de arquivos. Mas, pelos e-mails e comentários que recebi, percebi que não foi suficiente para o pessoal ir além do básico. Por isso, nesse artigo explico como criar formulários para envio de vários arquivos e principalmente, como processar esse envio com o PHP.

O artigo está dividido da seguinte maneira:
1) Introdução
2) Montando o formulário
3) Processamento do upload
4) Conclusão

1) Introdução

Praticamente em qualquer tipo de site e sistema há a necessidade de se fazer o upload de algum tipo de arquivo, principalmente imagens e documentos.

Vamos tomar como exemplo um site de uma agência promotora de eventos. Esse site tem as seções Galeria de Fotos de Eventos (vamos chamar de Galeria) e Envio de Currículos (também vamos chamar apenas de Currículos).

O painel de controle possui o cadastro de conteúdo para cada uma dessas seções:
- **Galeria**: nome do evento, data, local e fotos.
- **Currículos:** nome do candidato, curriculo e foto de rosto.

CASO 1) **Quantidade variável de arquivos a serem enviados.** Cada evento obviamente possui um número variável de fotos. Agora como fazê-lo? Enviar uma foto de cada vez? Ou tirar o fato que são várias fotos e limitar apenas uma foto por evento, e assim, falar para o cliente que por limitações tecnológicas eles devem apenas tirar uma foto em cada evento (o que seria absurdo, rs.)? O principal foco do artigo é baseado nesse caso.

CASO 2) **Upload variável de arquivos de vários tipos.** Agora na parte de currículos, poderá ser enviado currículo do usuário em múltiplos formatos (.doc, .pdf, etc), além de enviar a foto. Mudando apenas algumas linhas primeiro caso, esse caso fica concluído.

Pré-requisitos: para ler esse artigo, você já deve ter conhecimento do processando de formulários e envio de arquivos com o PHP.
Caso não saiba, tenha dúvidas ou queira apenas dar uma refrescada leia meus 3 artigos:
1) Manipulando dados de Formulário com PHP – Parte 1
2) Manipulando dados de Formulário com PHP – Parte 2 (Upload de arquivos)
3) Upload de Imagens com Segurança

2) Montando o formulário

Para apenas um campo de formulário, usa-se o nome que bem entender. Para vários arquivos, também, mas adiciona-se **`[]`** (abre e fecha colchetes) no final do nome, transformando o campo de upload em um array de campos. Parece complicado? Mas não é, pois tenho certeza que você já está careca de fazer isso. É o mesmo que acontece com um checkbox e selectbox de múltiplas opções (ver Manipulando dados de Formulário com PHP – Parte 1).

Também não esquecer que upload de arquivos só é possível via método POST e que sempre deve ser colocado o atributo `enctype="multipart/form-data"` na declaração do formulário.

Para criar os múltiplos campos de upload, vamos para o CASO 1, em que a quantidade é variável. São necessárias então duas ações para gerar o formulário: definir a quantidade de campos e imprimí-los dentro de um formulário.

A melhor maneira é colocar tudo numa mesma página, mas como isso é um artigo e precisamos ser ágeis, vamos dividir em dois arquivos: form_conta.php – irá pedir a quantidade de campos e form_gera.php – irá gerar o formulário.

Vamos ao código de ambos e logo após as explicações.

form_conta.php

```
<form action="form_gera.php" method="post">
<b>Envio das fotos</b><br />
Qual a quantidade de imagens do Evento?<br /><br />
<input type="text" name="quantidade" size="5"/><br />
<input type="submit" value="OK"/>
</form>
```

form_gera.php

```
<?php

// Obtém quantidade enviada. Perceba que é verificado se foi
fornecido um número inteiro,
// caso contrário é usada uma quantidade padrão, 5.
$Quantidade = (isset($_POST['quantidade']) && is_int(intval($_POST['quantida
de']))) ? (int)$_POST['quantidade'] : 5;

// Abre formulário de upload
echo '<form action="processa_upload.php" method="POST" enctype="multipart/fo
rm-data">';
echo '<b>Envio das fotos</b><br />';

// Imprime os campos para upload, de acordo com a quantidade pedida
for($i = 1; $i <= $Quantidade; ++$i)
{
    echo 'Foto #' . $i . ': <input type="file" name="fotos[]" /><br/>';
}

// Fecha formulário
echo '<br /><input type="submit" value="OK"/>';
echo '</form>';

?>
```

- Após você enviar seu formulário pedindo dados comuns (não é mostrado aqui, porque é o pré-requisito para que você leia esse artigo), no nosso caso o nome do evento, local e data, deve ser chamado **form_conta.php**. Nada mais é que um campo `text` de formulário que pede para o usuário digitar um número, no caso a quantidade de fotos a ser enviada.

- **form_conta.php** chama **form_gera.php**, que imprimirá o nosso formulário de. O ponto chave é o loop que itera até completar a quantidade fornecida, imprimindo os campos para upload da foto. Deve-se notar o nome dos campos é sempre o mesmo, **`fotos[],`** dessa maneira, o PHP saberá que estará recebendo vários arquivos no upload (será criado um array chamado $_FILES['fotos']).

- E por último, **form_gera.php** chama **processa_upload.php**, que como o nome diz, irá processar a validação e o envio dos arquivos. É o que veremos na próxima seção.

3) Processamento do upload

O nosso arquivo de processamento do formulário é o **processa_upload.php**. O que ele faz é obter o campo de upload, que é um array, e iterar sobre esse array fazendo a validação e o envio arquivo por arquivo, de acordo com o que estiver no array:

processa_upload.php

```
<?php

A) // Pasta de destino das fotos
$Destino = './eventoxyz/fotos/';
B) // Obtém dados do upload
$Fotos = $_FILES['fotos'];
// Contagem de fotos enviadas
$Conta = 0;
```

```php
C) // Itera sobre as enviadas e processa as validações e upload
for($i = 0; $i < sizeof($Fotos); $i++)
{
    D) // Passa valores da iteração atual
    $Nome    = $Fotos['name'][$i];
    $Tamanho = $Fotos['size'][$i];
    $Tipo    = $Fotos['type'][$i];
    $Tmpname = $Fotos['tmp_name'][$i];

     // Verifica se tem arquivo enviado
    if($Tamanho > 0 && strlen($Nome) > 1)
    {
        E) // Verifica se é uma imagem
        if(preg_match('/^image\/(gif|jpeg|jpg|png)$/', $Tipo))
        {
            // Caminho completo de destino da foto
            $Caminho = $Destino . $Nome;

            F) // Tudo OK! Move o upload!
            if(move_uploaded_file($Tmpname, $Caminho))
            {
                echo 'Foto #' . ($i+1) . ' enviada.<br/>';

                // Faz contagem de enviada com sucesso
                $Conta++;
            }
            else // Erro no envio
            {
                // $i+1 porque $i começa em zero
                echo 'Não foi possível enviar a foto #' . ($i+1) . '<br/>';
            }
        }
    }
}

if($Conta) // Imagens foram enviadas, ok!
{
    echo '<br/>Foi(am) enviada(s) ' . $Conta . ' foto(s).';
}
else // Nenhuma imagem enviada, faz alguma ação
{
    echo 'Você não enviou fotos!';
}
?>
```

A) Primeiro é definida a pasta para onde irão as fotos.

B) Obtém-se o array de fotos enviadas, passando para a variável `$Fotos`. O ponto chave é que o PHP transforma um campo de formulário `nome[]` em um array de sub-arrays para cada um dos indíces (name, type, size, tmp_name e error).

Exemplificando:
a) Quando é apenas um campo, digamos, `documento`, o PHP cria o array:
`$_FILES['documento']`, que por sua vez é um array que contém os índices:
- name, size, tmp_name, type e error.

b) **O PONTO PRINCIPAL DO ARTIGO:** Quando é um campo array, `fotos[]`, e digamos que tenham sido enviadas 2 fotos, o PHP cria um array $_FILES['fotos'], contendo os mesmo índices de sempre, mas cada indice desse é um novo array, cada uma com a informação de cada uma das fotos:

```php
$_FILES['fotos'] = array(
    'name'     => array('nome1.jpg', 'nome2.jpg'),
    'size'     => array(12345, 53223),
    'tmp_name' => array('dir/bla/img.tmp', 'dir/bla/img2.tmp'),
    'error'    => array(0, 0)
```

```
);
```

Deu para entender? Quando são múltiplos envios a variável que identifica o campo de upload é a mesma quando é um arquivo, acontece que cada índice é um novo array, de acordo com a quantidade de campos de upload. Se eu quero saber o nome e tamanho do segundo arquivo, simplesmente atuo num array normalmente:

```
$_FILES['fotos']['name'][0];
$_FILES['fotos']['size'][0];
```

C) Como as informações do upload estão num array, é iterado normalmente por ela (usando um loop for) como se fosse um outro array qualquer!

D) Para cada índice, é passado o valor da iteração para novas variáveis: `$Nome, $Tamanho, $Tipo` e `$Tmpname`, que contêm informações diretas sobre o arquivo, como se fosse o upload de um só arquivo:

```
$Foto['name']  = nome do arquivo
$Foto['size']  = tamanho em bytes do arquivo
$Foto['type']  = MIME Type do arquivo
$Foto['tmp_name']  = local do upload, para usar na função move_uploaded_file
$Foto['error'] = código de erro, não usado nesse artigo
```

E) Verifica se é uma imagem (através da checagem de MIME/TYPE – caso queira enviar um documento, basta mudar esse tupo de checagem). Assim, caso "aprovada" é copiada para a pasta desejada (em **F**).

Assim, move-se para a próxima foto (próximo item do array `$Fotos`).

ATENÇÃO: Note que só entra nos próximos passos, caso verificado que tenha sido enviado um arquivo naquele campo, através do if (verificando tamanho e nome) :

```
if($Tamanho > 0 && strlen($Nome) > 1)
```

Pois como é um array de campos, se forem 10 campos, mas o usuário colocar algum arquivo em só 2 deles, de qualquer forma o PHP retornará um array com 10 itens. Por isso, é importante certificar de que realmente foi enviado arquivo naquela iteração.

Por exemplo: se o usuário enviar uma foto no campo 1 e somente outra no campo 5, os dados serão enviados corretamente devido à checagem de envio.

4) Conclusão

Para o caso 2 (envio de documentos e uma foto), é o mesmo processamento contido em **processa_upload.php**, com a diferença que em **D** em vez de checar por imagem, coloque para checar por tipo de documento, mude:

```
if(preg_match('/^(.*)\.(doc|txt|pdf|xls|htm|html|rtf)$/', $Nome))
```

E após o loop de processamento dos documentos, para enviar a foto do usuário, simplesmente coloque um processamento simples de upload (conforme estão carecas de saber!).

Usem as informações contidas nos meus 4 artigos sobre formulário e upload e poderão fazer um sistema eficiente e seguro.

É isso! O próximo artigo que tratará de formulários, falará sobre o envio em tempo real dos dados usando AJAX, bem como upload de fotos e documentos com AJAX.
E estou preparando um artigo que ensina a criação de uma galeria de fotos completa. Fiquem antenados!

O código fonte desse artigo pode ser baixado clicando aqui.

Até o próximo!
Qualquer dúvida, basta me contatar!
Alfred Reinold Baudisch
alfred@auriumsoft.com.br
www.auriumsoft.com.br/blog/

Auriumsoft – Inteligência, Tecnologia e Vídeo
www.auriumsoft.com.br

AuriumHost – Hospedagem de qualquer tecnologia e banco de dados!
www.auriumhost.com.br