

Aula 10

Professor:

Inês Dutra

Escritas a bancos de dados e
validação no cliente e servidor

PHP: Escritas em bancos de dados

- Bancos de dados são utilizados não somente para consultas, mas também para **armazenar** novas informações
- Armazenamento de dados requer uma série de técnicas diferentes das usadas para leitura
- Por exemplo, noções de transações e concorrência são necessárias

PHP: Escritas em bancos de dados

- Objetivos deste módulo:
 - ▶ Técnicas básicas para lock e unlock de tabelas
 - ▶ Como implementar escritas em bancos de dados de forma segura para que vários usuários possam acessar o banco de dados simultaneamente

PHP: Escritas em bancos de dados

- Atualização, Inserção e Remoção de dados
- Antes de estudar técnicas de escrita em bancos de dados, vamos estudar o problema de *reload* de páginas
- Situação: form para entrada de dados do usuário, onde a entrada é transformada em uma escrita ao banco de dados
- Problema: escrita poderá ser realizada várias vezes, de forma indesejada, se o usuário clicar várias vezes o botão *reload* ou *refresh*

PHP: Escritas em bancos de dados

- Este problema (de *reload*) pode acontecer em várias outras situações:
 - ▶ Imprimindo páginas
 - ▶ Salvando o url no navegador e voltando à página utilizando um *bookmark* ou favoritos
 - ▶ Utilizando os botões *Backward and Forward*
 - ▶ Clicando <enter> na janela do url
 - ▶ Redimensionando a janela do navegador
- POST é menos suscetível a este problema, porque, após a primeira atualização do banco de dados, se o usuário quiser recuperar o script, o navegador pergunta ao usuário se quer re-enviar os dados (REPOST). Se o usuário responder OK, o problema ocorrerá.

PHP: Escritas em bancos de dados

- Em alguns casos, por exemplo, atualização de tabelas, *reload* não é um problema: a tabela vai ser atualizada sempre com os mesmos valores
- Porém precisamos de uma solução geral

PHP: Escritas em bancos de dados

- Solução: utilização de redireção de notificação
 - ▶ Usuário submete um form com variáveis e valores para uma submissão de escrita a um banco de dados
 - ▶ A operação SQL é executada no servidor
 - ▶ Falhando ou não, um cabeçalho (header) é enviado para o navegador instruindo-o a redirecionar para uma nova página, de notificação
 - ▶ Uma página informativa, porém sem código que possa causar algum efeito colateral, é exibida ao usuário

PHP: Escritas em bancos de dados

- Exemplo:

```

<?php
    include "error.inc";
    include "clean.inc";
    if (empty($_POST["titulo"]) || empty($_POST["descricao"]))
    {
?>
        <html>
        <head>
            <title> Inserção de novos dados </title>
        </head>
        <body>
            <form enctype="multipart/form-data"
                  action="slide_p8.php" method="post">
                Novo título:
                <br><input type="text" name="titulo" size=80>
                <br> Descrição:
                <br> <textarea name="descricao" rows=4 cols=80>
                </textarea>
                <input type="hidden"
                      name="MAX_FILE_SIZE" value="100000">
                <br> Capa (formato GIF):
                <input name="capa" type="file">
                <br><input type="submit">
            </form> </body> </html>

```

script 'slide_p8.php'

[Abrir editor](#)

[Abrir navegador](#)

script 'error.inc'

[Abrir editor](#)

script 'clean.inc'

[Abrir editor](#)

PHP: Escritas em bancos de dados

- Exemplo:

```

<?php
}
else
{
    $titulo = clean($_POST["titulo"],50);
    $descricao = clean($_POST["descricao"],2048);
    $capa = $_FILES["capa"]["tmp_name"];
    if (!($con = @mysql_connect($_SERVER["REMOTE_ADDR"],"aluno","aluno")))
        die("Não pôde conectar ao banco de dados");
    if(!mysql_select_db("prog2",$con))
        ExibeErro();
    if (is_uploaded_file($capa))
    {
        $arquivo = fopen($capa,"r");
        $arqconts = fread($arquivo,filesize($capa));
        $arqconts = AddSlashes($arqconts);
    }
    $arqconts = NULL;
    $inseredoado = "INSERT INTO livros (titulo,descricao,fotocapa) VALUES
                    ("."\"$titulo\".",."\"$descricao\".",".
                    "\"$arqconts\".");
    if (@mysql_query($inseredoado,$con)) && @mysql_affected_rows() == 1)
        header("Location: notificacao.php?". "idlivro=".mysql_insert_id($con) . "&status=T");
    else
        header("Location: notificacao.php?". "status=F");
}
?>

```

script 'slide_p8.php'

PHP: Escritas em bancos de dados

- Pontos importantes deste programa:
 - ▶ Utilização da função **header**
 - Passa uma string que é um cabeçalho HTTP (info sobre cabeçalhos HTTP pode ser encontrada em www.php.net)
 - No nosso caso, **string** é **Location:**, que indica ao navegador para onde redirecionar
 - Junto com **Location**, passa um valor de variável, **status**
 - ▶ Permite ao usuário fazer *upload* de um arquivo no formato GIF
 - ▶ Utiliza a função **mysql_insert_id()**;

PHP: Escritas em bancos de dados

```

<html>
<head> <title>Notificação</title>
</head>
<body bgcolor="white">
<?php
    include "error.inc";
    include "clean.inc";
    $host = $_SERVER["REMOTE_ADDR"];
    $idlivro = clean($_GET["idlivro"],4);
    $status = clean($_GET["status"],2);
    switch ($status) // operação foi bem sucedida?
    {
        case "T": // operação bem sucedida, apenas mostra res
            $consulta = "SELECT * FROM livros WHERE idlivro=".$idlivro;
            if (!$con = @mysql_connect($host,"aluno","aluno")))
                die("Não pôde conectar ao BD");
            if (!mysql_select_db("prog2",$con))
                ExibeErro();
            if (!($res = @mysql_query($consulta,$con)))
                ExibeErro();
            if ($row = @mysql_fetch_array($res))
            {
                echo "O seguinte título foi adicionado ao BD";
                echo "\n<br><img src=\"displayimage.php?idlivro=$idlivro\">";
            }
            break;
        case "F":
            echo "A operação de inserção falhou!";
            break;
        default:
            echo "Erro inesperado"
    } ?> </body> </html>

```

script 'notificacao.php'

[Abrir editor](#)

PHP: Escritas em bancos de dados

- Programa anterior implementa uma página de notificação ao usuário
- Recebe como parâmetro a variável **\$status** passada pela função **header**
- Quando **\$status = T**, faz uma consulta ao BD e devolve os detalhes da operação realizada
- O novo título inserido é identificado através da variável **\$idlivro**
- Quando **\$status = F**, o script exibe uma mensagem de falha

PHP: Escritas em bancos de dados

- Outro ponto importante:
 - ▶ Dados de arquivo inseridos no BD
- Arquivos são transferidos usando, de `<form>`:
 - ▶ o tipo `multipart/form-data`,
 - ▶ o método `POST`,
 - ▶ `input` de tipo `file`

PHP: Escritas em bancos de dados

- Exemplo

```
<form enctype="multipart/form-data"
        action="slide_p8.php"
        method="POST">
<br> Capa (formato GIF):
<input name="capa" type="file">
<br><input type="submit">
</form>
```

PHP: Escritas em bancos de dados

- Recuperação do arquivo armazenado no BD:

```
<?php  
...  
$dado = @mysql_fetch_array($res);  
if (!empty($dado["fotocapa"]))  
{  
    header("Content-Type: image/gif");  
    echo $dado["fotocapa"];  
}  
?>
```

script 'displayimage.php'

script 'displayimage.php'

Abrir editor

PHP: Escritas em bancos de dados

- O script anterior é chamado pelo script **notificacao.php** utilizando um tag html ****

```
"<img src=\"displayimage.php?idlivro=". $idlivro ."\">";
```

- Neste caso, o script **notificacao.php**, além de exibir informações gerais sobre o item incluído no BD, também exibe uma imagem associada ao item

PHP: Escritas em bancos de dados

- Técnicas utilizadas para arquivos de imagens pequenas
- São necessárias configurações adicionais para arquivos grandes:
 - ▶ **MAX_FILE_SIZE**, como no exemplo
 - ▶ Limite de memória do script PHP deve ser maior do que o tamanho do arquivo. Pode ser ajustado com o parâmetro **memory_limit** no arquivo **php.ini** (normalmente localizado em **/usr/local/lib/**)
 - ▶ Tamanho máximo de arquivos para upload pode ter que ser modificado (**upload_max_filesize** no arquivo **php.ini**)
 - ▶ O tamanho máximo de POST deve ser maior do que o tamanho do arquivo (**post_max_size** em **php.ini**)
 - ▶ O tempo máximo de execução do script deve ser um valor apropriado para que haja tempo suficiente para fazer upload do arquivo. O default é de 30 segundos. Pode ser modificado em **max_execution_time** de **php.ini**
 - ▶ O servidor web deve ser re-inicializado após as modificações para que **php.ini** possa ser recarregado (**apachectl restart**)

PHP: Escritas em bancos de dados

- Função utilizada na inserção, remoção ou atualização de tabelas:
 - ▶ `mysql_affected_rows`
- Função utilizada na inserção de dados em tabelas:
 - ▶ `mysql_insert_id`

PHP: Escritas em bancos de dados

- `int mysql_affected_rows ($con)`
- Retorna o número de linhas de tabela afetadas pelo último comando UPDATE, DELETE ou INSERT da consulta
- Não pode ser usada com SELECT
- `mysql_num_rows` é usada com SELECT, como mencionado em outras aulas
- `mysql_affected_rows` retorna `0` se o registro já tiver sido removido ou se não existir

PHP: Escritas em bancos de dados

- `mysql_insert_id`
- Retorna o valor do identificador AUTO_INCREMENT associado com o INSERT mais recente
- É usada, por exemplo, para retornar o valor de `idcliente` quando clientes são criados utilizando AUTO_INCREMENT

PHP: Escritas em bancos de dados

- Inserção de dados
 - ▶ 3 fases
 - Entrada de dados
 - Validação
 - Inserção

PHP: Escritas em bancos de dados

- Primeira fase:

```

<?php
if (! (isset ($_POST["nome"]))) {
?>
<html>
<head><title>Informações para Cadastro</title></head>
<body bgcolor="white">
<form method="post" action="slide_p22.php">
<h1> Informações para Cadastro </h1>
<h3>Por favor, entre com a informação solicitada para se cadastrar. Campos mostrados em
<font color="red"> vermelho </font> são obrigatórios.</h3>
<table>
<col span="1" align="right">
<tr> <td><font color="red">Sobrenome:</font></td>
    <td><input type="text" name="sobrenome" size=50></td></tr>
<tr> <td><font color="red">Nome:</font></td>
    <td><input type="text" name="nome" size=50></td></tr>
<tr> <td><font color="red">Endereço:</font></td>
    <td><input type="text" name="endereco" size=50></td></tr>
<tr> <td><font color="red">Cidade:</font></td>
    <td><input type="text" name="cidade" size=50></td></tr>
<tr> <td><font color="red">Data de nascimento (dd/mm/yyyy) :</font> </td>
    <td><input type="text" name="ddn" size=10></td></tr>
<tr> <td><font color="red">Email/username:</font></td>
    <td><input type="text" name="email" size=50></td></tr>
<tr> <td><input type="submit" value="Submit"></td></tr>
</table></form></body>
</html>

```

script 'slide_p22.php'

[Abrir editor](#)
[Abrir navegador](#)

PHP: Escritas em bancos de dados

- Segunda fase:

```

<?php
} // fecha if
else {
    include 'error.inc';
    include 'clean.inc';

$errorString = "";
foreach($HTTP_POST_VARS as $varname => $value)
    $formVars[$varname] = trim(clean($value, 50));
if (empty($formVars["nome"]))
    $errorString .= "\n<br>O campo nome deve ser digitado.";
if (empty($formVars["sobrenome"]))
    $errorString .= "\n<br>O campo sobrenome deve ser digitado.";
if (empty($formVars["endereco"]))
    $errorString .= "\n<br>Você deve digitar pelo menos uma linha de endereço.";
if (empty($formVars["cidade"]))
    $errorString .= "\n<br>O campo cidade é obrigatório.";
if (empty($formVars["ddn"]))
    $errorString .= "\n<br>Você deve preencher sua data de nascimento.";
elseif (!ereg("^(0[1-9]|1[0-2])/(0[1-9]|1[0-2])/([0-9]{4})$", $formVars["ddn"], $partes))
    $errorString .= "\n<br>A data não está no formato correto: DD/MM/YYYY";
if (empty($formVars["email"]))
    $errorString .= "\n<br>O campo email é obrigatório.";
if (!empty($errorString))
{
    // Há erros. Mostrar e sair.
?>
...continua no Próximo slide

```

PHP: Escritas em bancos de dados

- Segunda fase: script 'slide_p22.php'

```
<html>
<head><title>Erro no cadastramento</title></head>
<body bgcolor="white">
<h1>Erro no Cadastramento</h1>
<?=$errorString?>
<br><a href="slide_p22.php">Retornar para o form</a>
</body>
</html>
<?php
    exit;
}

// Se o script chegou aqui, é porque os dados foram recebidos e tratados com sucesso
// Terceira fase!
```

PHP: Escritas em bancos de dados

```
if (!($con = @ mysql_pconnect($hostName, $username, $password)))
    die("Não pode conectar ao BD");
```

script 'slide_p22.php'

```
if (!mysql_select_db($databaseName, $connection))
    ExibeErro();

// Formatar data de acordo com formato interno Mysql
$ddn = " \"$partes[3]-$partes[2]-$partes[1]\"";
```

```
// Criar uma consulta com o dado do usuário
$consulta = "INSERT INTO cadastro
    set id = 0, "
    "sobrenome = \" . $formVars["sobrenome"] . "\", "
    "nome = \" . $formVars["nome"] . "\", "
    "endereco = \" . $formVars["endereco"] . "\", "
    "cidade = \" . $formVars["cidade"] . "\", "
    "email = \" . $formVars["email"] . "\", "
    "ddn = $ddn";
```

```
// Rodar a consulta
if (!(@ mysql_query ($consulta, $con)))
    ExibeErro();
// Id do novo membro
$ID = mysql_insert_id();
// Redirecionar usuário
header("Location: notificaco_cadastro.php?ID=$ID");
} % end else
?>
```

script 'notificacao_cadastro.php'

Abrir editor

PHP: Escritas em bancos de dados

- Problema de escritas: concorrência de leitores e escritores
- Situações que devem ser evitadas em escritas a bancos de dados:
 - ▶ Problema da atualização perdida (*lost update*)
 - ▶ Problema da leitura de dados "sujos" (*dirty read*)
 - ▶ Problema de incorreção de sumários (*incorrect summary*)
 - ▶ Problema de leitura não determinística (*unrepeatable read*)

PHP: Escritas em bancos de dados

- Atualização perdida:
 - ▶ 2 usuários A e B
 - ▶ A lê do BD
 - ▶ B lê do BD e atualiza
 - ▶ A atualiza por cima da atualização de B, sem saber que B atualizou

PHP: Escritas em bancos de dados

- Leitura de dados "sujos":
 - ▶ 2 usuários A e B
 - ▶ A lê um dado e atualiza
 - ▶ B lê o novo valor e atualiza
 - ▶ Por algum motivo, A resolve que a transação deve ser desfeita
 - ▶ Problema: B já trabalhou com o dado atualizado por A

PHP: Escritas em bancos de dados

- Incorreção de sumários:
 - ▶ 2 usuários A e B
 - ▶ A está fazendo atualizações ao mesmo tempo em que B está lendo os valores para fazer um sumário dos dados
 - ▶ Resultado imprevisível para B, dado que as leituras e atualizações podem ocorrer em qualquer ordem temporal

PHP: Escritas em bancos de dados

- Leitura não determinística:
 - ▶ 2 usuários A e B
 - ▶ A lê um valor
 - ▶ B atualiza
 - ▶ A volta a ler o valor para verificação: valor não é mais o mesmo!!

PHP: Escritas em bancos de dados

- Solução: utilização de **locks**
- O que são locks?
 - ▶ Variáveis especiais que permitem acesso exclusivo de leitura e/ou escrita a um arquivo ou tabela
 - ▶ Lock de leitura: permite que vários usuários possam ler uma tabela, porém impede vários escritores
 - ▶ Lock de escrita: permite apenas um leitor e um escritor para uma tabela

PHP: Escritas em bancos de dados

- Exemplo em MySQL:

```
mysql> LOCK TABLES itens READ, rel_temp WRITE;
mysql> SELECT sum(preco) FROM itens WHERE id=1;
+-----+
| sum(preco) |
+-----+
|      438.65 |
+-----+
1 row in set (0.04 sec)
mysql> UPDATE rel_temp SET total=438.65 WHERE id=1;
mysql> UNLOCK TABLES;
```

PHP: Escritas em bancos de dados

- Regras de uso:
 - ▶ Se um usuário deseja **escrever** em uma tabela e está executando uma transação suscetível a problemas de concorrência, deve adquirir um **lock de escrita** para aquela tabela
 - ▶ Se um usuário deseja apenas **ler** uma tabela e está executando uma transação suscetível a problemas de concorrência, deve adquirir um **lock de leitura** para aquela tabela
 - ▶ Se um usuário requisita um lock, deve adquirir lock para **todas** as tabelas utilizadas na transação
 - ▶ O usuário deve liberar todos os locks quando a transação estiver completa

PHP: Escritas em bancos de dados

- Locks para escrita têm prioridade sobre locks para leitura
- Pode levar a situações de *starvation*, onde leitores são sempre postergados e nunca têm uma chance de obter o lock
- Na prática, aplicações lêem mais dados do que escrevem em tabelas, portanto esta situação é rara
- Prioridade mais baixa pode ser dada ao lock de escrita, se necessário.
Ex:

```
LOCK TABLES ... LOW_PRIORITY WRITE
```

PHP: Escritas em bancos de dados

- Atenção: não confundir INSERT DELAYED de MySQL com operações de lock
- INSERT DELAYED utilizado apenas para adiar inserção de dados no BD sob controle do MySQL
- Não garante exclusividade de escrita ou leitura como um lock

PHP: Escritas em bancos de dados

- Locks também podem ser utilizados para obter maior desempenho no acesso a bancos de dados
- Por exemplo, se uma determinada consulta requer urgência em coletar dados de um BD e este está sendo compartilhado por muitos usuários, podemos usar LOCK TABLES ... WRITE para garantir acesso exclusivo ao BD e, desta forma, obter os resultados mais rapidamente
- Garante:
 - ▶ Dedição de todo o sistema e recursos à consulta em questão
 - ▶ Acesso mais rápido ao disco
- Desvantagem: pouca concorrência no servidor e usuários pouco satisfeitos com a lentidão da resposta ou timeouts

PHP: Escritas em bancos de dados

```

<?php
include 'error.inc';
function AtualizaDesconto($idCliente, $IdItem, $desc, $min, $con)
{
    $ok = false;
    // Lock todas as tabelas envolvidas nesta transação
    $cons = "LOCK TABLES itens READ, compras WRITE, cliente READ";
    if (!mysql_query($cons, $con))
        ExibeErro();
    // Quanto o usuário gastou até o momento?
    $cons = "SELECT SUM(preco*qtde) FROM itens, compras, cliente
            WHERE cliente.id = compras.idCliente AND compras.IdItem = itens.IdItem
            AND itens.idCliente = compras.idCliente AND compras.idItem = $IdItem
            AND cliente.id = $idCliente";
    if (!$result = mysql_query($cons, $con)) ExibeErro();
    $row = mysql_fetch_array($result);
    // A qtde comprada é maior do que um mínimo?
    if ($row["SUM(preco*qtde)"] > $min) // se sim, dar desconto
    {
        $cons = "UPDATE compras SET desc = $desc WHERE idCliente = $idCliente AND IdItem = $IdItem";
        if (!mysql_query($cons, $con)) ExibeErro();
        $ok = true;
    }
    $cons = "UNLOCK TABLES"; // Unlock tabelas
    if (!mysql_query($cons, $con)) ExibeErro();
    return $ok; // Retorna se desconto foi concedido ou não
}

```

script 'slide_p37.php'

[Abrir editor](#)

[Abrir navegador](#)

PHP: Escritas em bancos de dados

script 'slide_p37.php'

```
// MAIN -----
if (!($con = @ mysql_connect ($hostName, $username, $password)))
    die("Não pode conectar ao DB");
if (!mysql_select_db($databaseName))
    ExibeErro();
// Dá um desconto de $4.95 ao cliente 653 no item #2, se gastou mais de $10
$desc = AtualizaDesconto(653, 2, 4.95, 10, $con);
?>
<html>
<head>
    <title>Exemplo do uso de locks</title>
</head>
<body bgcolor="white">
<?php
    if ($desc == true)
        echo "<h3>Desconto dado</h3>";
    else
        echo "<h3>Sem desconto</h3>";
?>
</body>
</html>
```

PHP: Escritas em bancos de dados

- Alguns cuidados a serem tomados ao utilizar locks:
 - ▶ Garantir que todas as tabelas acessadas no contexto de um lock foram "locked" com o comando LOCK TABLES. O script dá erro se alguma tabela for acessada no contexto de um lock e não estava na lista de LOCK TABLES
 - ▶ Lembrar sempre de usar UNLOCK TABLES quando a transação terminar. LOCK TABLES-UNLOCK TABLES devem ser usados no mesmo script
 - ▶ Todos os scripts que requisitam locks devem ter a estrutura: lock, consulta, atualiza, unlock
 - ▶ Importante usar UNLOCK TABLES em conexões persistentes que tenham adquirido um lock

PHP: Escritas em bancos de dados

- Por default, MySQL usa tabela do tipo MyISAM que não suporta outros níveis de lock
- Outros níveis de locking que podem ser utilizados em MySQL:
 - ▶ Nível de disco (Berkeley Database, BDB)
 - ▶ Nível de linha de tabela (InnoDB)
 - ▶ Nível de linha de tabela e tabela (Gemini)
- Suporte para estes tipos de bancos de dados é obtido através da compilação de MySQL durante a instalação

PHP: Validação no servidor e no cliente

- Validação de dados é essencial em aplicações web que usam bancos de dados para manter integridade e consistência do banco de dados
- 3 ambientes possíveis para validação: BD, cliente e servidor
- Na camada cliente a validação ocorre no navegador e normalmente é feita em dados provenientes de um form. Javascript é normalmente utilizada com esta finalidade
- Validação no lado do servidor é normalmente feita através de um script que roda na camada intermediária, como já visto anteriormente

PHP: Validação no servidor e no cliente

- Vamos estender o que já foi discutido anteriormente, quando utilizamos clean e empty

PHP: Validação no servidor e no cliente

- Validação consiste em dois procedimentos
 - ▶ Encontrar o erro
 - ▶ Apresentar mensagens de erros
- Encontrar o erro pode ser uma tarefa:
 - ▶ Interativa: dados são verificados a medida em que são digitados
 - ▶ Pós-validação: dados são verificados após terem sido todos digitados
- Apresentação de mensagens pode ser:
 - ▶ Campo a campo: uma nova msg de erro é exibida para cada campo digitado erradamente
 - ▶ Batched: todos os erros são apresentados em uma única mensagem
- Além disso, podemos utilizar outras dimensões: nível do erro, experiência do usuário etc.

PHP: Validação no servidor e no cliente

- 4 abordagens comuns:
 - ▶ Validação interativa campo a campo
 - ▶ Validação interativa com erros em batch
 - ▶ Pós-validação com erros campo a campo
 - ▶ Pós-validação com erros em batch
- Já utilizamos pós-validação com erros em batch

PHP: Validação no servidor e no cliente

- Modelos interativos são difíceis de implementar em um ambiente web
- Scripts que rodam no servidor podem ficar sobrecarregados com validação interativa de dados
- Validação no lado do cliente pode ser implementada utilizando o modelo interativo, porém validação apenas no lado do cliente pode levar a falhas de segurança

PHP: Validação no servidor e no cliente

- Modelos de pós-validação são preferidos em aplicações web que lidam com bancos de dados
- Ambos cliente e servidor podem validar dados de forms durante o processo de submissão do form
- Validação duplicada no cliente e no servidor é essencial por causa da falta de confiabilidade dos scripts no lado do cliente

PHP: Validação no servidor e no cliente

- O modelo de pós-validação pode ser combinado com apresentação de erros campo a campo ou em batch
- O modelo batch é mais adequado no servidor pois não atrapalha o desempenho
- O modelo campo a campo tem um overhead maior e é mais lento porque cada erro do form requer uma solicitação e resposta HTTP

PHP: Validação no servidor e no cliente

- No lado do cliente que usa pós-validação, ambos os modelos para apresentação de erros podem ser usados
- Vantagem do campo a campo: cursor é posicionado diretamente no campo que está errado
- Desvantagem: vários erros geram várias mensagens de erros, que podem frustrar o usuário

PHP: Validação no servidor e no cliente

- Vantagem do modelo batch: todos os erros são apresentados em uma única mensagem
- Desvantagem: o cursor não é redirecionado para o campo que precisa de correção
- Escolha do modelo de relatório de erro depende do tamanho e da complexidade do form e dos requisitos do sistema

PHP: Validação no servidor e no cliente

- Regras gerais para validação:
 - ▶ Telefones e faxes devem ser numéricos e estar de acordo com um *template* conhecido (já fizemos exemplos destes anteriormente)
 - ▶ Endereços de email devem estar de acordo com os requisitos disponíveis em <http://www.ietf.org>
 - ▶ Nomes de domínios devem existir
 - ▶ Datas devem obedecer o formato padrão adotado na aplicação

PHP: Validação no servidor e no cliente

- Elementos que normalmente devem ser validados:
 - ▶ Endereços de email
 - ▶ Datas
 - ▶ Nomes de domínios
 - ▶ Idade
 - ▶ etc

PHP: Validação no servidor e no cliente

- Exemplo de validação de email

```
$valEmailExpr = "[0-9a-zA-Z!#$%&_-.]{1}[.][0-9a-zA-Z!#$%&_-.}]*";  
if (empty($formVars["email"]))  
    $errorString .= "Campo email não pode estar em branco";  
elseif (!eregi($valEmailExpr,$formVars["email"]))  
    $errorString .= "O endereço de email deve estar no formato  
                    name@domain.format";  
elseif (strlen($formVars["email"]) > 50)  
    $stringError .= "O endereço de email não pode exceder 50 caracteres";  
elseif (!getmxrr(substr(strstr($formVars["email"], '@'), 1), $temp))  
    || checkdnsrr(gethostbyname(substr(strstr($formVars["email"], '@'), 1), "ANY"))  
$errorString .= "Este domínio não existe";
```

PHP: Validação no servidor e no cliente

- Utilizando Javascript para validar dados de entrada

```
<html>
<head>
<title>Exemplo simples em Javascript</title>
<script type="text/javascript">
<!-- escondendo o script de navegadores mais antigos
function contembrancos(s)
{
    for (var i=0; i &lt; s.value.length; i++)
    {
        var c = s.value.charAt(i);
        if ((c==' ') || (c=='\n') || (c=='\t'))
        {
            alert("Campo não deve conter espaços em branco");
            return false;
        }
    }
    return true;
} // fim do comentário --&gt;
&lt;/script&gt;</pre>
```

script 'slide_p53.php'

Abrir editor

Abrir navegador

PHP: Validação no servidor e no cliente

- Utilizando Javascript para validar dados de entrada

```
</head>                                         script 'slide_p53.php'  
<body>  
<h2> Form para entrada de username </h2>  
<form onSubmit="return (contembrancos(this.userName));"  
method="post" action="teste.php">  
  <input type="submit" value="SUBMIT">  
</form>  
</body>  
</html>
```

script 'teste.php'

Abrir editor