INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Campus São João da Boa Vista

Relatório Técnico

4º ano – Curso Técnico em Informática

Prof. Breno Lisi Romano

Criação e Desenvolvimento de Páginas Dinâmicas do Modulo de Reclamações do Projeto Reclame são João

Aluno: Gabriel Pella Nogueira

Prontuário: 142047x

São João da Boa Vista – SP

2017

Resumo

Informo através deste documento os benefícios de se utilizar páginas dinâmicas em um projeto, entre muitos ows benefícios talvez o principal seja a economia gerador por este simples procedimento, que consiste em manter poucas consultas no banco de dados e fazer com que as informações retornadas sejam armazenadas em variáveis e manipuladas através das mesmas

Sumário

1 Introdução

Esse Relatório técnico traz os benefícios da criação de páginas dinâmicas no modulo de reclamações do projeto Reclame São João.

Este projeto tem o objetivo de fazer a aproximação dos estabelecimentos da cidade de São João da Boa Vista com os seus clientes, permitindo que os mesmos avaliem e entrem em contato com os estabelecimentos em questão para levar até eles problemas que, se resolvidos, melhorem essa relação de cliente empresa e consequentemente aumentem as vendas da empresa em questão.

O projeto foi desenvolvido pelos alunos do Instituto Federal (IFSP) Campus São João da Boa Vista, com apoio da prefeitura municipal.

A utilização de páginas dinâmicas faz com que muita memória e rede sejam poupadas, fazendo com que o site seja mais rápido.

2 Desenvolvimento

Páginas dinâmicas são aquelas onde o código é processado dentro da mesma, ou seja, quando algum usuário a acessa o código é carregado e executa tudo que é requisitado, entre os benefícios talvez o mais importante seja o carregamento rápido da página, pois com a pressa dos usuários tempo é essencial

```php
$telefone = null;
$link     = $_GET["link"];
$busca    = $_GET["EST_NOME"];
$idatual = 0;
$estabelecimentosPdo = new EstabelecimentosDAO;
$resultado = $estabelecimentosPdo->obterEstabelecimentoPorNome($busca);

if (count($resultado)==0) {
    echo "<script>alert('Estabelecimento nao encontrado')</script><script>window.location='$link';</script>";
} else {



        foreach ($resultado as $row) {
            $nome_responsavel            = $row["EST_NOME_RESPONSAVEL"];
            $total_reclamacao            = $row["EST_TOTAL_RECLAMACAO"];
            $nome_fantasia               = $row["EST_NOME_FANTASIA"];
            $id_estabelecimento          = $row["EST_ID"];
            $numero_endereco             = $row["EST_NUMERO_ENDERECO"];
            $publico_alvo                = $row["EST_PUBLICO_ALVO"];
            $facebook_empresa            = $row["EST_FACEBOOK_EMPRESA"];
            $longitude                   = $row["EST_LONGITUDE"];
            $latitude                    = $row["EST_LATITUDE"];
            $bloqueio                    = $row["EST_STATUS_BLOQUEIO"];
            $media_avaliacao_consumidores = $row["EST_MEDIA_AVALIACAO_CONSUMIDORES"];
            $media_reclamacao            = $row["EST_MEDIA_RECLAMACAO"];
            $complemento                 = $row["EST_COMPLEMENTO"];
            $site                        = $row["EST_SITE_EMPRESA"];
        }
```

Figura 1 - Código em arquivo rec_estabelecimentos Projeto Reclame São João

No exemplo anterior, temos a consulta no banco de dados pela variável "$estabelecimentosPdo" e através de um foreach (para cada item retornado) foram atribuídos os dados que foram retornados a cada uma das variáveis, fazendo com que não seja necessária uma nova consulta

```
<div class="estrelas" style="margin-bottom: 0%;">
<label for="cm_star-1"><i class="fa"></i></label>
<input type="radio" id="cm_star-1" name="fb" value="1" <?php
if ($historico_ava[0]["AVA_NOTA"] >= 0 && $historico_ava[0]["AVA_NOTA"] < 2) {
$ranking = "cm_star-1";
}
?>/>
<label for="cm_star-2"><i class="fa"></i></label>
<input type="radio" id="cm_star-2" name="fb" value="2" <?php
if ($historico_ava[0]["AVA_NOTA"] >= 2 && $historico_ava[0]["AVA_NOTA"] < 3) {
$ranking = "cm_star-2";
}
?>/>
<label for="cm_star-3"><i class="fa"></i></label>
<input type="radio" id="cm_star-3" name="fb" value="3"<?php
if ($historico_ava[0]["AVA_NOTA"] >= 3 && $historico_ava[0]["AVA_NOTA"] < 4) {
$ranking = "cm_star-3";
}
?>/>
<label for="cm_star-4"><i class="fa"></i></label>
<input type="radio" id="cm_star-4" name="fb" value="4"<?php
if ($historico_ava[0]["AVA_NOTA"] >= 4 && $historico_ava[0]["AVA_NOTA"] < 5) {
$ranking = "cm_star-4";
}
?>/>
<label for="cm_star-5"><i class="fa"></i></label>
<input type="radio" id="cm_star-5" name="fb" value="5"<?php
if ($historico_ava[0]["AVA_NOTA"] >= 5 && $historico_ava[0]["AVA_NOTA"] < 6) {
$ranking = "cm_star-5";
}
?>/>
```

Figura 2 - Código em arquivo rec_estabelecimentos Projeto Reclame São João

Na figura 2 (dois) podemos ver as variáveis sendo manipuladas, este procedimento de armazenar dados em variáveis faz com que seja economizado muitos milissegundos de processamento pois ao invés de uma nova consulta, estamos apenas manipulando as variáveis onde eles foram armazenados

2.1 Segurança

Além da antes mencionada economia de memória e o ganho de tempo de se usar variáveis para armazenar retornos do banco, temos também a ganho na área de segurança de informação, uma área ignorada por muitos desenvolvedores por "tomar muito tempo", eles não entendem que um simples teste de segurança pode poupar várias horas e milhares de reais de prejuízo por um ataque hacker.

O principal problema que essa técnica resolve é o SQL INJECTION, que consiste em uma falha de programação onde o usuário permite que uma variável get ou post seja inserida diretamente no banco de dados, como por exemplo;

SELECT * FROM table WHERE id= '$_POST ["variavel_post"]'.

Isso permite que o hacker insira uma variável a seu gosto fazendo com que seja retornado algo indesejado

Uma forma de impedir isso é substituir a variavel_post por uma variável carregada na própria página como por exemplo:

$variavel = $_POST ["variavel_post"]

Isso anularia a invasão de um hacker sem muito conhecimento, mas para a segurança total de uma página, sempre deve-se usar PDO (PHP Data Objects) que faz com que se o acesso ao banco seja bem mais seguro.

3. Conclusão

Como pudemos observar os benefícios de se utilizar páginas dinâmicas em um projeto são realmente notáveis e deveriam ser uma exigência de todos os gerentes de projetos, pois a não utilização do mesmo tem graves consequências em dinamicidade e na segurança da página.

Alem de não tomar muito tempo do programador, essas simples mudanças podem fazer de um site lento e com pouco trafego, um site rápido com muitos acessos fazendo com que, se for o caso de um site com objetivos financeiros, sua receita aumente muito.