02

Salvando no banco de dados

Transcrição

Anteriormente, vimos como adicionar *meta boxes*, pois percebemos que a nossa página de imóveis continha poucas informações. Assim, incluímos novos campos a serem preenchidos e editados, com preço do imóvel, vagas de garagem, banheiros e quartos. Lançamos a função `add_meta_box()`, para o qual passamos uma série de definições. Entendemos que, para `add_action()`, primeiramente não passamos `init`, e sim `add_meta_boxes`, pois não é possível chamar a função logo no início da renderização do Wordpress.

Vimos os códigos HTML e CSS colocados dentro do nosso formulário, sendo o CSS *inline* pois, como não temos acesso ao *head*, não conseguimos importá-lo. Poderíamos isolá-lo em uma função, porém, não iremos fazê-lo neste caso. Agora, lidaremos com a questão de salvar as informações que incluirmos nos campos recém implementados, que atualmente não aparecem em lugar algum.

Primeiramente, precisaremos que elas sejam salvas no banco de dados. E quem faz a comunicação com o banco de dados é o próprio Wordpress, que possui uma forma de atualizar os metadados das postagens, a função `update_post_meta()`. É necessário levar em consideração que estamos lidando com imóveis, mas eles são derivados de post, e o Wordpress, no fim das contas, trata tudo como sendo tal, uma vez que um imóvel é o nosso *custom post type* que criamos.

Digamos que queiramos atualizar o dado do preço, por exemplo, então, precisaremos passar uma chave acerca deste dado para o Wordpress. No entanto, incluir simplesmente `preco_id` não indicará valor algum. Se inspecionarmos a página do dashboard, e buscarmos pela tag `<form>`, notaremos que o Wordpress envia estes dados para `post.php`, que é trazido por `action`.

Na URL, veremos que a página atual é o `post.php`, isto é, o Wordpress envia os dados para a página em que estamos, então, precisaremos pegá-los em `functions.php`, como variáveis. Quais são estas variáveis que enviam informações via URL? GET ou POST, na maioria dos casos, certo? Verificamos isto por meio de `method="post"`, e teremos que indicar seu campo, que no caso é o próprio `preco_id`.

Ainda, salvaremos as demais informações:

```
add_action('add_meta_boxes', 'registra_meta_boxes');

update_post_meta('preco_id', $_POST['preco_id']);
update_post_meta('vagas_id', $_POST['vagas_id']);
update_post_meta('banheiros_id', $_POST['banheiros_id']);
update_post_meta('quartos_id', $_POST['quartos_id']);
```

Para organizarmos melhor o nosso código, como de costume, isolaremos o trecho em uma função. E as informações serão atualizadas quando os posts forem salvos, ou seja, `save_post`. A variável `$_POST['preco_id']` está sendo recebida, mas o que será salvo se ela não existir? Precisamos garantir que isto seja salvo apenas quando `preco_id` existir na variável POST, e então faremos isso com todos os campos a serem preenchidos.

```
function atualiza_meta_info() {
    if( isset($_POST['preco_id']) ) {
        update_post_meta('preco_id', $_POST['preco_id']);
```

```
    }
    if( isset($_POST['vagas_id']) ) {
        update_post_meta('vagas_id', $_POST['vagas_id']);
    }
    if( isset($_POST['banheiros_id']) ) {
        update_post_meta('banheiros_id', $_POST['banheiros_id']);
    }
    if( isset($_POST['quartos_id']) ) {
        update_post_meta('quartos_id', $_POST['quartos_id']);
    }
}

add_action('save_post', 'atualiza_meta_info');
```

Entretanto, mesmo com toda esta implementação, ao submetermos um valor no campo referente a preço e clicarmos em "Atualizar", nenhum dado será mostrado, o que não quer dizer que nossos dados não estejam sendo salvos. Para confirmarmos isto, acessaremos `localhost/phpmyadmin` pelo navegador, o gerenciador do nosso banco de dados, para utilizarmos esta ferramenta.

Clicaremos na aba "Databases > wp_malura", e verificaremos uma série de tabelas, dentre as quais a que nos interessa é `wp_postmeta`. Vamos até as últimas linhas desta tabela, onde se encontram `preco_id` na coluna "meta_key", que é a chave, com "200000" em "meta_value", o valor. Se os dados estão de fato sendo guardados no banco de dados, falta exibi-los em algum lugar, como em nosso *input*, mesmo.

Em `functions.php`, temos nossos `<input>`s em relação a estes campos, e para incluir um valor dentro deles, utilizaremos `value`, que terá um código PHP.

```
<input id="maluras-preco-input" class="maluras-metabox-input" type="text" name="preco_id" value=
```

Para isto, será necessário capturarmos o valor e salvá-lo em uma variável a ser criada (`imoveis_meta_data`), que receberá `get_post_meta()`, uma função do Wordpress que serve para capturar as metainformações acerca de um post específico. No entanto, não estamos passando post nenhum, e esta função deve receber o ID, o número do post, para que se possa fazer a query no banco de dados. Porém, em vez de deixarmos `$post->ID`, existe uma forma de injetarmos o post diretamente na função.

Em `informacoes_imovel_view()`, teremos:

```
function informacoes_imovel_view( $post ) {
    $imoveis_meta_data = get_post_meta( $post->ID );
    ?>

    // restante do código omitido
}
```

Em seguida, voltaremos ao trecho que estávamos alterando para incluirmos `$imoveis_meta_data`, que retornará um array com `preco_id`, que é a *meta key*, isto é, a chave cujo valor precisamos. Entretanto, como não guardamos várias posições, e sim somente uma, poderemos acrescentar `[0]`.

```
<input id="maluras-preco-input" class="maluras-metabox-input" type="text" name="preco_id" value=
```

Salvaremos o arquivo e testaremos preenchendo os campos destinados ao preço, quantidade de vagas, banheiros e quartos no imóvel, no dashboard, e clicando no botão "Atualizar". O valor será exibido como gostaríamos, porém os demais números desaparecerão, pois não definimos para que eles fossem impressos na tela. Corrigiremos isto replicando o trecho `value="<?= $imoveis_meta_data['preco_id'][0]; ?>"` e aplicando-o aos demais `<input>` s, substituindo cada `preco_id` por `vagas_id` , `banheiros_id` e `quartos_id` , respectivamente.

Ao testarmos, teremos que tudo funciona conforme esperado. Continuando, o campo destinado ao preço está formatado de um jeito estranho, não? Para melhorarmos esta formatação, incluiremos a função `number_format()` no `value` do primeiro `<input>` , o de preço, sendo que o primeiro parâmetro indicará o que será formatado de fato, o segundo, a quantidade de casas decimais (`2`), e o terceiro, os separadores a serem utilizados (`,` e `.`).

```
<input id="maluras-preco-input" class="maluras-metabox-input" type="text" name="preco_id" value=
```

Agora, reparem que `$_POST['preco_id']` está vindo "cru", ou seja, independentemente do que o administrador for digitar para salvar nestes campos, isto será salvo em nosso banco de dados. Algo como apóstrofos, as chamadas "aspas simples", poderiam ser digitadas, confundindo o banco de dados, ou caracteres especiais, e assim por diante.

Para evitar tais problemas de *strings*, os desenvolvedores do Wordpress criaram **funções *sanitize***, para "limpar" as *strings*. Usaremos `sanitize_text_field()` :

```
function salvar_meta_info_imoveis( $post_id ) {
    if( isset($_POST['preco_id']) ) {
        update_post_meta( $post_id, 'preco_id', sanitize_text_field($_POST['preco_id']) );
    }
    if( isset($_POST['vagas_id']) ) {
        update_post_meta( $post_id, 'vagas_id', sanitize_text_field($_POST['vagas_id']) );
    }
    if( isset($_POST['banheiros_id']) ) {
        update_post_meta( $post_id, 'banheiros_id', sanitize_text_field($_POST['banheiros_id'])
    }
    if( isset($_POST['quartos_id']) ) {
        update_post_meta( $post_id, 'quartos_id', sanitize_text_field($_POST['quartos_id']) );
    }
}
```

Feito isso, voltaremos ao dashboard, atualizaremos a página, e teremos tudo funcionando da maneira desejada.