Bir Eklenti Hikâyesi / WordPress

İbrahim Altunok

Nisan, 2016

Her şey sitemizin ana sayfasına haber ekleme konusunda bir kolaylık oluşturmak uğruna başladı. Bu kolaylığı sağlayacak bir eklenti lazımdı bize, şöyle forumdaki iletinin adresini biz gireceğiz, o da iletiyi WordPress yazısı olarak kendisi ekleyecek şekilde. Kolları sıvayıp işe koyuldum. Daha önce iletişim paneli, katalog/ürün yönetimi gibi WordPress'in kendisine müdahale etmeyen kendi hâlinde eklentiler yazmıştım; ama böyle bir amaçla ilk defa eklenti yazıyordum, dolayısıyla ilk başta zorlanacağımı düşünüyordum. Ancak eklentiyi yazmaya başladıkça WordPress ile bu işlerin ne kadar kolay olduğunu anladım. Ve ortaya tek sayfalık, 200'e yakın satırlık (boş satırlar ve yorum satırları da dahil) bir kod çıktı. Bu yazıyı da gerek eklenti yazımında ipuçları içermesi için, gerekse dergi okuyucularına eklentiyi dağıtabilmeme bahane olması için yazayım, dedim.

Kabaca eklentinin özellikleri şunlar: WordPress Yazı Ekle sayfasına bir metin kutusu ve bir düğme içeren bir kutucuk ekliyor ve metin kutusuna ileti adresini yazıp düğmeye bastığımızda, iletinin başlığını ve içeriğini sayfada ilgili kısımlara ekliyor. Daha sonra siz yazıda son düzenlemeleri yapıp kaydediyorsunuz. Aslında ortada çok büyük bir emek ve benzersiz bir ürün yok. Ama dediğim gibi, maksat eklentiyi paylaşmak, paylaşırken de dergiye katkı yapmış olmak sadece. (Anlatımın bundan sonraki kısmı temel düzeyde HTML ve PHP bilgisi, yeterli miktarda da jQuery bilgisi gerektirmektedir.)

Önce kodun genel çatısını anlatıp sonra aralarını doldurmak daha mantıklı olur sanırım. Eklentimiz altı adımdan oluşacak.

1. Oluşturacağımız kodun bir eklenti olacağını WP'ye söylemek
2. WP Yazı Ekle sayfasına bir kutu yerleştirmek
3. Bu kutunun içeriğini oluşturmak
4. Forum ileti içeriğini çekecek olan ayıklayıcı (parser) kodu yazmak
5. Ayıklayıcı koddan gelen bilgileri WP Yazı Ekle sayfasında uygun kısımlara yerleştirmek
6. Yazı kaydedilirken iletinin forumdaki adresini kaydedecek bir fonksiyon yazmak

Bu adımları gerçekleştirecek olan kod çatısı da şu şekilde:

```php
<?php

/* eklenti bilgileri */

add_action( 'add_meta_boxes', 'forumhaber_kutu_ekle' );
add_action( 'save_post', 'forumhaber_kaydet' );
add_action( 'wp_ajax_forumhaber_ayikla', 'forumhaber_ayikla' );
if ( in_array( $pagenow, array('post.php', 'post-new.php') ) ) {
    add_action( 'admin_head', 'forumhaber_js' );
}

function forumhaber_kutu_ekle() {
    add_meta_box(
    /* Kutu bilgileri */
    );
}

function forumhaber_kutu_icerigi( $post ) {
    /* Kutu içeriği (HTML olarak) */
}

function forumhaber_js(){
    /* Ajax fonksiyon içeriği (jQuery olarak) */
}

function forumhaber_ayikla() {
    /* Ayıklayıcı PHP kodu */
}

function forumhaber_kaydet( $post_id ) {
    /* WP'de yazı kaydedilirken tetiklenecek fonksiyonumuz */
}
?>
```

Gördüğünüz gibi oldukça basit bir eklenti olacak eklentimiz. İçeriği doldurmadan önce WordPress ile ilgili birkaç bilgi aktarmam gerekiyor.

WordPress'in gücünü aldığı özelliklerinden biri "kanca"larıdır (hook). Kancalar belli etiketlere sahiptir ve WP'de sayfa oluşturulurken ilgili etikete sahip kancalara tutunan fonksiyonlar tetiklenir. WP'de iki tür kanca vardır. Birisi "eylem" kancası (action hook) diğeri de "süzgeç" kancası (filter hook) ismini almaktadır. (Yazının bundan sonraki kısmında gerek olduğunda eylem, süzgeç ve kanca kelimelerini kullanacağım.) WP'de bir kancaya bir fonksiyon tutturmak için iki hazır fonksiyon kullanılır. "add_action" ve "add_filter". Bu iki fonksiyon, elzem olan iki parametre kabul eder, ilki tutulacak kancanın etiketi, diğeri de tutunacak fonksiyonun ismi. Eylem ve süzgeç kancalarının farkı ise şudur; eylemler tetiklendiğinde sadece bir fonksiyon çalışır ve biter, süzgeçler tetiklendiğinde ise önce tutunduğu kancanın etiketi ile ilgili bir değişken alır ve o değişken üzerinde bir değişiklik yaptıktan sonra değişkeni geri döndürmek zorundadır.

Bir de çeviri mevzusu var tabii ki. WP'de (aslında genelde) çeviri için kullanılan birkaç hazır fonksiyon vardır. Bunlardan ilki iki adet alt çizgiden oluşan fonksiyon, bir diğeri de bir alt çizgi ve 'e' harfinden oluşan fonksiyon. (Oluşan derken isminden bahsediyorum.). Yani

```
1 __( 'çevrilecek metin', 'çeviri isim alanı');
```

ve

```
1 _e( 'çevrilecek metin', 'çeviri isim alanı');
```

İlk fonksiyon, metni çevirdikten sonra çevrilmiş hâlini bir değişken olarak geri döndürür. Diğeri ise metni çevirdikten sonra doğrudan ekrana basar. Çeviri isim alanı (text domain) dediğimiz parametre kullanılmak zorunda değildir, kullanılmadığı takdirde WP'nin ana çeviri dizinine göre çevrilir. Kullandığınız zaman ise eklentiniz için oluşturulacak olan çeviri dizinine göre çevrilir. Bu parametreyi kullanmak için bir çeviri isim alanı oluşturmak zorunda da değilsiniz. Bu size, ileri geliştirme evresi için çevrilebilir bir eklenti oluşturmuş olmanıza yarar.

Son olarak da kullanılacak değişkenlerde ve fonksiyonlarda dikkat edilmesi gereken bir hususta bahsedeceğim. Malum olduğu üzere WordPress, etkin olan tüm eklentilere ve temaya ait olan dosyaları okuduktan sonra sayfayı oluşturmaya başlar. Durum böyle olunca da diğer eklentilerle çakışma olmaması için fonksiyonlarımızda ve değişkenlerimizde benzersiz isimler kullanmak zorundayız. Ben bu eklenti için "forumhaber" önekini tercih ettim.

Şimdi başlayabiliriz.

PHP sayfamızın eklenti olduğunu WordPress'e anlatarak başlıyoruz

Bildiğiniz gibi her WordPress eklentisi şu satırlarla başlamak zorundadır. Eklenmediği takdirde WP bu PHP sayfasının bir eklenti olduğunu anlayamaz ve sonuç olarak eklentiyi kullanamazsınız. Bu satırlar eklentiye ait bilgileri içerir.

```
1  <?php
2  /*
3  Plugin Name: Forum Haber Ekleyici
4  Plugin URI: http://www.ubuntu-tr.net
5  Description: Forumdaki iletinin sadece URL'sini girerek içeriğini almaya yarayan bir eklenti
6  Version: 1.0
7  Author: İbrahim Altunok
8  Author URI: http://www.ubuntu-tr.net
9  License: GPLv2
10 */
```

WordPress Yazı Ekle sayfasına eklentimizin kutusunu yerleştirelim

WordPress'te Yazı Ekle/Düzenle sayfası oluşturulurken, o gördüğümüz kutuların eklenmesi esnasında tetiklenen bir eylem vardır. Daha doğrusu WP'nin her köşesinde bir şekilde tetiklenen eylem ve süzgeçler vardır. Bu kısımda, yani bizim işimize yarayacak kısımda tetiklenecek olan eylem kancasının etiketi de "add_meta_boxes" etiketi. Ekrana ihtiyacımız olan kutuyu yerleştirecek fonksiyonumuzu bu kancaya tutturuyoruz.

```
1 add_action( 'add_meta_boxes', 'forumhaber_kutu_ekle' );
```

Bu eylem ekleme fonksiyonu ile “add_meta_boxes” eylemine bizim oluşturacağımız “forumhaber_kutu_ekle” fonksiyonumuzu iliştirmiş olduk. Yazı Ekle/Düzenle sayfasına kutular yerleştirilirken çağrılacak olan “forumhaber_kutu_ekle” fonksiyonumuzda ekrana bir kutu da biz yerleştireceğiz.

```
function forumhaber_kutu_ekle() {
  add_meta_box(
    'forumhaber_kutu',
    __( 'Forum İleti Bilgileri Alanı', 'forumhaber_textdomain' ),
    'forumhaber_kutu_icerigi',
    'post',
    'normal',
    'default'
  );
}
```

Bu fonksiyon ile ekrana bir kutu yerleştirmiş olduk. Bunun için de WP’nin hazır fonksiyonlarından olan “add_meta_box” fonksiyonunu kullandık. Bu fonksiyonda ilk parametremiz kutumuzun kimliği (ID), ikincisi kutu başlığında yazılacak yazı (Gördüğünüz üzere bunu çeviriye uygun şekilde yazdım.), üçüncüsü kutunun içeriğini oluşturacağımız fonksiyonumuzun ismi, dördüncüsü bu kutunun hangi sayfaya ekleneceği, beşincisi ve altıncısı da kutunun konumuyla alakalı parametreler. Bunlardan bazılarına açıklık getirmek gerekecek.

Kutunun ekleneceği sayfa derken, bu parametre iki adet değer alabilir; birisi “post”, diğeri “page”. Bu iki değer tanıdık gelmiştir. WordPress’te girdiler iki çeşittir; biri “yazı (post)” diğeri de “sayfa (page)”. Bizim eklentimiz için “post” değeri yeterli olduğundan “page” değerini dahil etmedim. WP’nin Sayfa Düzenleme sayfasında da kutumuzun görünür olması için bu fonksiyonları aynı parametrelerle bir kez daha yazmanız gerekir. İkincide sadece “post” değerini “page” şeklinde yazmak yeterlidir.

Kutumuzun yerleşeceği konum için, fonksiyonu yazarken “normal” ve “default” değerlerini kullandık. WP Yazı Ekle sayfasını gözünüzde şöyle canlandırın; soldaki sütunu üst ve alt iki bölmeye ayrılmış olan, iki sütunlu bir sayfa. Sol üst bölmenin karşılığı “normal”, sol alt bölmenin karşılığı “advanced”, sağ sütunun karşılığı ise “side”. Bu üç bölmeden hangisine yerleştireceğinize karar verdikten sonra önceliğini belirleyeceksiniz. Bu belirtilen konumlarda, kutular şu sıraya göre yerleşir : “high” » “core” » “default” » “low” (Ayrıntısı için şu resme bakabilirsiniz : http://www.wproots.com/wp-content/uploads/2011/08/positions.png)

Kutu içeriğini oluşturalım

Yukarıda belirttiğimiz gibi kutumuzun içini dolduracak fonksiyonumuzun ismini “forumhaber_kutu_icerigi” olarak seçtik. Bu fonksiyonda yapacaklarımız şu şekilde; öncelikle o an düzenlenmekte olan bir yazı varsa o yazıyla ilgili bilgimizi alacağız, daha sonra form elemanlarımızı yerleştireceğiz. Sonra da eklentimize has bir özellik olarak bu kutuyu sayfanın en başına yerleştireceğiz.

Bildiğiniz gibi etiketlerinin içerisindeyken sadece “echo” ve muadili olan fonksiyonlarla ekrana yazı yazılabilir. Ama bu etiketlerin dışına çıkıldıktan sonra konulmuş olan her türlü şey doğrudan ekrana yazı olarak gider. Biz de HTML ve JS türündeki içeriği ekrana yazarken her seferinde “echo” fonksiyonuyla uğraşmak yerine PHP etiketinden çıkıp normal bir HTML sayfası hazırlar gibi kodlarımızı yazacağız.

```
function forumhaber_kutu_icerigi( $post ) {

  $forumhaber_url = get_post_meta($post->ID, 'forumhaber_url', true);
  wp_nonce_field( plugin_basename( __FILE__ ), 'forumhaber_noncename' );

  ?>

  <div style="padding:10px 0px;">
    <label for="forumhaber_url_alan">
      <?php _e("Bağlantı", "forumhaber_textdomain" ); ?>
    </label>

    <input type="text" id="forumhaber_url" name="forumhaber_url" value="<?php echo
        $forumhaber_url;?>" size="50" />

```

```
15      <input type="button" id="forumhaber_parse" value="<?php _e('İçeriği Al','
            forumhaber_textdomain');?>">
16
17      <img src='images/wpspin_light.gif' id='forumhaber_yukleniyor' style='display:none'>
18    </div>
19
20    <script type="text/javascript">
21
22    (function($) {
23
24      if($("#forumhaber_kutu")) $("#titlediv").prepend($("#forumhaber_kutu"));
25
26    })(jQuery);
27
28    </script>
29
30    <?php
31  }
```

İlk iki satırı ayrıntılı açıklayalım.

```
1  $forumhaber_url = get_post_meta($post->ID, 'forumhaber_url', true);
```

Üstte de dediğim gibi eğer yeni bir yazı değil de mevcut bir yazıyı düzenliyorsak, "add_meta_box" fonksiyonu bizim fonksiyonumuzu çağırırken düzenlenmekte olan yazıyı da parametre olarak gönderir. Biz de bu parametreyi alıp eğer daha önce bu yazının numarasına kayıtlı olan bir URL bilgisi var mı yok mu diye bakıyoruz. Varsa bunu ekrana yerleştireceğimiz metin kutusunun içerisine koyacağız. Bu bakma işlemini de yine WordPress'e ait hazır fonksiyonlardan biri olan "get_post_meta" fonksiyonunu kullandık. Bu fonksiyonun ilk parametresi yazı numarası (yani "post id"), ikinci parametresi de alınacak bilginin anahtarı (meta key) oluyor. Üçüncü parametre ise fonksiyonun döneceği bilginin türü ile alakalı. Yani eğer (varsayılan olan) false olarak ayarlanırsa dönecek veri bir dizi (array) olur. Eğer bizim yaptığımız gibi true olarak ayarlanırsa da tek bir sonuç döner ve o da "string" türünde olur. Bilgi anahtarı dediğim şeyi (kaydetme ile ilgili olan fonksiyonumuzun içerisinde) biz belirliyoruz.

```
1  wp_nonce_field( plugin_basename( __FILE__ ), 'forumhaber_noncename' );
```

Bu satır ise WordPress'te kullanılan güvenlik ögelerinden birine ait. Bu fonksiyon sayfaya "hidden" özelliğindeki bir form elemanı yerleştirir. Görevi, yazıyı kaydederken kullanacağımız fonksiyonda kaydedilecek değerlerin bu sayfadan geldiğini anlamamızı sağlamaktır. (Nonce : Number used Once) Kısaca "nonce" şu şekilde bir yapıdır, bir form oluşturulurken, sadece bir kez üretilip kullanılabilecek bir değer oluşturulur. Bunun için kullanılan şeyler genelde şunlardır: O anki kullanıcıya özel olan bir değer, mesela PHP Session ID değişkeni, sürekli değişebilen bir değer, mesela Unix Timestamp, bir de kontrol için kullanılacak sabit bir değişken, mesela WP'nin de bu fonksiyonunda bizden istediği ilk parametrede olduğu gibi eklentimizin dosya yolu. Kayıt fonksiyonumuzda dosya yolumuzu kontrol amaçlı kullanarak form bilgisinin bize harici olarak değil de bu dosyada oluşturulmuş bir şekilde geldiğini anlayacağız. İkinci parametre ise yerleştirilecek olan form elemanının "name" özelliği olarak ayarlamaya yarar.

Daha sonra PHP etiketinden çıkıp HTML olarak form elemanlarımızı ekliyoruz. Burada HTML'nin ayrıntısına fazla girmeyeceğim, ancak bir iki noktaya da değinmeden geçemeyeceğim. Dikkatinizi çekmiştir, form elemanları ekliyoruz ama bir form eklemiyoruz; çünkü tüm kutuları kapsayan bir form WordPress tarafından sayfaya yerleştirilir. Kaydetme esnasında da bu bilgiler her kutunun "kaydetme" eylemine bağlı fonksiyona iletilir. Bizim kaydetme fonksiyonumuz en sonda gelecek.

"Name" özelliği "forumhaber_url" olan metin kutumuzun "value" özelliğine, ilk satırda aldığımız $forumhamber_url değişkeninin değerini yazıyoruz. Eğer yeni bir yazı oluşturuyorsanız bu işlem yazınızı etkilemez; çünkü bu durumda değişkenin bir değeri olmaz, yani içi boştur.

```
1  <img src='images/wpspin_light.gif' id='forumhaber_yukleniyor' style='display:none'>
```

"İçeriği Al" etiketli düğmemizin yanına bir de resim dosyası yerleştirdik. Bu resim, düğmeye basıldığında bizim "yükleniyor" canlandırmamız olacak. Doğal olarak sayfa ilk yüklendiğinde görünür olmaması lazım. Bunun için de "style" özelliğine "display:none" ifadesini ekleyerek onu "görünmez" kıldık. Düğmeye basıldığında görünür yapıp iş bittiğinde tekrar görünmez kılacağız.
u tutturacağız. Bunun da sadece yazı düzenleme sayfasında gerçekleştirilmesini istiyoruz. O yüzden "$pagenow" değişkenini kontrol ederek doğru sayfada olup olmadığımıza bakıyoruz. Doğru sayfadaysak da "admin_head" kancasına bir fonksiyon iliştiriyoruz. "admin_head" kancası, admin kanadında oluşturulmakta olan bir sayfanın

HTML etiketleri oluşturulurken tetiklenen eyleme aittir. Bu "head" etiketleri oluşturulurken WP bizim fonksiyonumuza da uğrayacak ve bizim JavaScript kodumuz admin sayfasında "head" etiketleri arasına yerleşmiş olacak. Şimdi de yerleşecek olan JS kodlarını yazalım.

```
function forumhaber_js(){
  ?>

  <script type="text/javascript">

  jQuery(document).ready(function($) {
    $("#forumhaber_parse").click(function(){
      $("#forumhaber_yukleniyor").show();
      var data = {
        action : 'forumhaber_ayikla',
        forumurl : $("#forumhaber_url").val()
      };
      $.get(ajaxurl, data, function(d){
        $("#title-prompt-text").hide();
        $("#title").val(d.baslik);

        if($("#content").css("display") != "none")
          $("#content").html(d.ileti);

        else
          $("#content_ifr").contents().find("body").html(d.ileti);

        $("#forumhaber_yukleniyor").hide();
      },'json');
    });
  });

  </script>

<?php
}
```

"forumhaber_parse" ifadesi, bizim "İçeriği Al" isimli düğmemizin kimliği, "forumhaber_yukleniyor" ise "yükleniyor" canlandırması olarak kullanacağımız "img" elemanının kimliği oluyor. Düğmeye tıklandığında çağrılacak fonksiyonun ilk satırında bu resim dosyasını görünür yapıyoruz.

Burada anlatmam gereken en önemli konu WordPress'te Ajax kullanımı. WordPress'te Ajax kullanmak çok basittir. Yapmamız gereken tek şey, "wp_ajax_" ifadesiyle başlayan ve bize özel olan bir eylem kancası belirlemek (Evet bu sefer kancanın etiketini biz belirliyoruz; ancak başında "wp_ajax_" ifadesi olmak şartıyla…) ve bu kancaya bir fonksiyon tutturmak. WordPress'te Ajax kullanımı amacıyla oluşturulmuş bir "ajax.php" dosyası bulunmaktadır. Bu dosya, normal "index.php" gibi tüm eklentileri (ve temayı) tarar, "wp_ajax_" ifadesiyle başlamış olan kancaları seçer, kendisine "action" indeksiyle (mesela "ajax.php?action =forumhaber_ayikla") gelen bilgiye uyan kancayı belirler, o kancaya tutunmuş olan fonksiyonu çalıştırır ve işini bitirmiş olur.

WordPress, kullanılacak olan "ajax.php" dosyasının yolunu sayfanın başında belirler ve "ajaxurl" isminde bir değişkene atar. Jquery'de Ajax için türetilmiş olan fonksiyonlardan "$.getfonksiyonunukullanyoruz.JqueryAjaxfonksiyonlarpara$
$...eklindeolanksmdabunlarbelirledik.(Bunu.$get" fonksiyonun içinde de yazabilirdik.)

Üçüncü parametre olarak da Ajax fonksiyonumuz başarıya ulaşırsa yapacağımız işler için bir fonksiyon yazıyoruz.

Eğer Ajax ile çalıştırılan dosya bize bir değer döndü ise bu değer jQuery tarafından bu yazdığımız fonksiyona parametre olarak gönderilir. (İsmi önemli değildir, ben kolaylık olsun diye sadece "d" yazdım.)

Ajax dosyasından ileti ile ilgili bilgiler geldiği zaman işlemlere başlıyoruz. İlk olarak WP'de başlığı girdiğimiz kutuda "Başlığı girin" şeklinde soluk bir şekilde görünen, "title-prompt-text" kimliğine sahip olan yazıyı gizliyoruz. Daha sonra başlık kutusuna, yani "title" kimliğine sahip olan metin kutusuna Ajax sayfasından dönen bilgi içerisinden aldığımız başlığı yazıyoruz. Sonra da Yazı Ekle sayfasındaki düzenleyicilerin içerisine ileti içeriğini ekliyoruz. (Bildiğiniz üzere iki düzenleyici var, HTML ve Görsel düzenleyici.) Görsel düzenleyici, esasında bir "iframe" elemanı olduğu için bu kısmı biraz dolambaçlı yapmak zorundayız. En sonunda da "yükleniyor" canlandırmamızı tekrar gizli hâle getiriyoruz.

Sıra geldi ayıklayıcı fonksiyonumuza

Yukarıda WordPress'te Ajax kullanımını anlatmış, kullanımın JS kanadını göstermiştim. Şimdi ise WP'de Ajax kullanımının PHP kanadını göstereceğim. Yapacağımız şey, "wp_ajax_forumhaber_ayikla" etiketi ile bir eylem kancası türetip yazdığımız fonksiyonumuzu bu kancaya tutturmak. Fonksiyon içerisinde de $_GET ile gelen URL bilgisini alıp bu adrese gidip iletinin içeriğini ayıklayacağız. Yalnız ayıklama kısmının teknik ayrıntılarına fazla girmeyeceğim maalesef.

```
1  add_action('wp_ajax_forumhaber_ayikla', 'forumhaber_ayikla');
2
3  function forumhaber_ayikla() {
4    if(!isset($_GET['forumurl']) || $_GET['forumurl'] == '') die();
5
6    $data = '';
7
8    if(!extension_loaded("curl")){
9      $ch = curl_init();
10     curl_setopt($ch, CURLOPT_URL, $_GET['forumurl']);
11     curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
12     curl_setopt($ch, CURLOPT_FOLLOWLOCATION, TRUE);
13     curl_setopt($ch, CURLOPT_HEADER, FALSE);
14     $data = curl_exec($ch);
15   }
16
17   else {
18     $data = file_get_contents($_GET['forumurl']);
19   }
20
21   preg_match('/msg(\w+)/', $_GET['forumurl'], $msgid);
22   $msg = $msgid[^1];
23
24   preg_match('#<a id="msg' . $msg . '"></a>.*?windowbg.*?>(.*?)<hr class="post_separator" />#
         si', $data, $div);
25   $div = $div[^1];
26   $div = preg_replace('#PHPSESSID=.*?&amp;#si', '', $div);
27
28   preg_match('#action=profile;u=(.*?)".*?>(.*?)</a>#si', $div, $user);
29   $username = $user[^2];
30   $userid = $user[^1];
31
32   preg_match('#<h5 id="subject_' . $msg . '">.*?<a.*?>(.*?)</a>.*?</h5>#si', $div, $title);
33   $return['baslik'] = str_replace("Ynt: ","",$title[^1]);
34
35   preg_match('#&\#171;(.*?)&\#187;#si', $div, $date);
36   $date = trim(preg_replace('#<strong>.*?</strong>#si', '', $date[^1]));
37   $date = preg_replace('#<b>.*?Bugün.*?</b>.*?,#si', date_i18n('d F Y') . " — ", $date);
38
39   $r=substr($div,strpos($div,'<div class="inner" id="msg_'.$msg.'">'));
40   $r2="";
41   for($d=0; ; $d++) {
42     $r2 = substr($r, 0, strpos($r, "</div>", $d) + 6);
43     preg_match_all('#<div#si', $r2, $ad);
44     preg_match_all('#</div#si', $r2, $kd);
```

```
45      if(count($ad[0]) == count($kd[0])) break;
46    }
47    $r2=substr($r2,strpos($r2,">")+1);
48    $r2=substr($r2,0,strrpos($r2,"</div>"));
49
50    $r2 = $r2 . "<br /><br />Bu ileti <i>".$date."</i> tarihinde " .
51      "<a href='http://forum.ubuntu-tr.net/index.php?action=profile;u=" .
52      $userid . "'><i>" . $username . "</i></a> " .
53      "tarafından yazılmıştır." .
54      "<br><a href='" . $_GET['forumurl'] . "' target='_blank'>" .
55      "İletiyi forumda açmak için tıklayınız »</a>";
56
57    $return['ileti'] = $r2;
58
59    echo json_encode($return);
60    die();
61  }
```

Oldukça karmaşık bir fonksiyon olduğunun farkındayım. Kabaca bu fonksiyonda neler olup bittiğinden, neler döndüğünden biraz bahsetmeye çalışayım. Öncelikle eğer $_GET değişkeni içerisinde "forumurl" değeri gönderilmemişse işleri baştan kesip atıyoruz. Daha sonra da sunucuda cURL paketinin kurulu olup olmadığına bakıp, kurulu ise cURL ile değilse PHP'nin yerleşik fonksiyonlarından olan file_get_contents fonksiyonu ile iletinin olduğu sayfanın HTML kaynak kodunu okuyoruz.

Önemli bir hususu burada belirtmek zorundayım. Bu fonksiyonun doğru bir şekilde, sağlıklı olarak çalışması için, girilen ileti adresinde "msg123456" şeklinde bir ifade bulunmak zorundadır. Çünkü içeriği okunacak olan iletinin olduğu kısmı, verilen adres bilgisinde bu kısmı okuduktan sonra elde ettiği o sayısal ifadeye göre ayıklayacak şekilde yazdım. Bu ayıklama kodu daha da pratikleştirilebilir, daha etkili yapılabilir veya başka siteler için sil baştan yazılabilir.

İletinin numarasını elde ettikten sonra yaptığımız şey, bir adet "başlık" bilgisi oluşturmak, bir adet de "ileti" (post) bilgisi oluşturmak. Aradaki kalabalığı iletinin altına ileti bilgilerini yerleştirebilmek için yazdım. Eğer başka bir site için bu eklentiyi düzenlemek isterseniz sadece bu fonksiyonu yeniden yazabilirsiniz. Sadece geri döndürdüğünüz değişkende "baslik" ve "ileti" indeksleriyle belirlenmiş bilgiler bulunması yeterli olacaktır.

Değer döndürmek derken bildiğiniz üzere Ajax kullanımı esnasında bilgi döndürmek demek ekrana bir şeyler yazmak demektir. Biz de bunu echo fonksiyonuyla yaptık. WordPress'te Ajax fonksiyonlarınızın işi bittiği zaman die() fonksiyonuyla sonlandırmanız, Ajax sürecinizin daha çabuk bitmesini sağlayacaktır.

Son olarak da kayıt fonksiyonumuzu oluşturalım

Artık eklentimizin neredeyse en kolay kısmına geldik ve bitiriyoruz. Bu adımda yapacaklarımız şunlar: Öncelikle WordPress'in yazı kaydetme esnasında tetiklediği eylemin kancasına fonksiyonumuzu tutturacağız. Bu kancanın etiketi "save_post". Daha sonra da kayıt işlemi için herhangi bir engel var mı yok mu kontrol edeceğiz; yoksa ileti adresini kaydedeceğiz.

```
1  add_action( 'save_post', 'forumhaber_kaydet' );
2
3  function forumhaber_kaydet( $post_id ) {
4
5    if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE )
6      return;
7
8    if ( !wp_verify_nonce( $_POST['forumhaber_noncename'], plugin_basename( __FILE__ ) ) )
9      return;
10
11   if ( 'post' == $_POST['post_type'] ) {
12     if ( !current_user_can( 'edit_post', $post_id ) )
13       return;
14   }
15   else
16     return;
17
```

```
18     update_post_meta($post_id, 'forumhaber_url', $_POST['forumhaber_url']);
19
20 }
```

Yaptığımız kontroller şunlar; eğer WordPress o esnada otomatik taslak kaydetme sürecindeyse biz kayıt yapmıyoruz, daha sonra yukarıda bahsettiğim Nonce kontrolünü yapıyoruz, yani bu URL bilgisi bizim sayfamızdan mı geliyor ona bakıyoruz, daha sonra kaydedilmekte olan şey bir "yazı" mı onu kontrol ediyoruz, yani kaydedilen şey "sayfa" ise biz oradan ayrılıyoruz, en son olarak da o anki etkin kullanıcının yazı kaydetme gibi bir yetkisi var mı yok mu ona bakıyoruz.

Eğer hiçbir sıkıntı yoksa artık iletinin adresini kaydedebiliriz. Bunun için kullanacağımız fonksiyon "update_post_meta". Bu fonksiyon, ismi itibariyle biraz yanıltıcı olabilir, sanki var olan bir şeyi güncelliyormuşuz gibi algılanabilir. Evet, bu fonksiyon aslında daha önce kaydedilmiş bir bilgiyi değiştirmeye yarar. Yeni bir bilgi kaydederken "set_post_meta" fonksiyonu kullanılır. Ancak "update_post_meta" fonksiyonu, bir avantaj olarak, eğer belirtilen anahtarda bir veri yoksa önce o anahtarı oluşturmaktadır. Yani bu fonksiyonu kullanmak için o anahtarın daha önce oluşturulmuş olmasına gerek yoktur. Bu özellik sayesinde, anahtar yoksa set_post_meta, varsa update_post_meta şeklindeki bir kontrolün hammallığından kurtarmış olur. Parametreleri de yazının numarası, bilgi anahtarı (meta key), bilgi verisi (meta value).

Hepsi bu kadar. Bu yazı ile sizlere WordPress için eklenti geliştirme hususunda birtakım ipuçları vermeyi amaçladım. Elbette ki WordPress ile yapılabilecekler bununla sınırlı değil. Gönül ister ki dilimizin döndüğünce tüm incelikleri aktarabilelim. Ancak şimdilik bu kadarla yetinmek durumundayız. Eklentiyi dergiyle birlikte indirebileceksiniz. Kolaylıkla kurup kullanmaya başlayabilirsiniz.

Buraya kadar okuyabildiğiniz için teşekkür ederim.