به نام ایزد دانا

ایجاد یک CMS با استفاده از با استفاده از برنامه نویسی شیئ گرا در PHP [1]

نویسنده: Peter Zeidman
مترجم: امیر حسین استخریان

بخش ۱

فرض کنیم شما می خواهید برای شبکه Interanetیک شرکت سایتی طراحی کنید ، پس احتمالا شما به یک CMS احتیاج خواهید داشت.(ابزاری برای سازمان دهی اسناد).برای سازمانهای تجاری لازم است آنها یک CMS منطبق بر نیازهای خاص خود آن شرکت داشته باشند

در این جا چگونگی ساخت یک CMS با استفاده از php و mysql شرح داده می شود . این سیستم امکان استفاده از پایگاه داده های دیگر مانند sql server و postgre sql را نیز فراهم می کند .
اولین و بهترین کار این سیستم امکان مدیریت و ذخیره سازی محتوای سایت داخلی ، یا سایت اینترنتی در یک پایگاه داده است .البته ویژگی های دیگر یمانند اعتبارسنجی کاربران و مدیریت فایل ها نیز برای اجرای موفق سایت ، الزامی است .
دانش اولیه ای از php برای کدنویسی CMS مورد نیاز است و فرض بر این است که یک سرور php و یک سیستم پایگاه داده ، در اختیار داریم .

[1] منبع : مجله رایانه شماره ۱۶۶**ایجاد یک CMS با استفاده از با استفاده از برنامه نویسی شیئ گرا در PHP**

قابلیت های این سیستم شبیه آن چیزی که در سیستم های post nuke و smarty یا سیستم های مدیریت محتوا وجود دارد نیست ، البته امکانات اولیه مورد نیاز یعنی مجموعه امکاناتی که به ما اجازه می دهد سیستم خود را ایجاد کنیم ، در این سیستم وجود دارد .

طراحی CMS

اولین گام در این مرحله تعیین مشخصات اولیه ای است که CMS باید داشته باشد . این مشخصات به نیازهای ما بستگی دارد :

- **مدیریت محتوا :** بهترین عملکرد این سیستم ، ذخیره سازی محتوا (اسناد) در پایگاه داده و نمایش آن ها به کاربران مطابق با درخواست آن ها می باشد . یک رابط کاربری که به کاربران امکان اضافه کردن ، حذف کردن و یا تغییر محتوا را می دهد نیز مورد نیاز است .
- **هویت سنجی کاربران (تشخیص هویت) :** بخش های خاصی از سایت داخلی یا سایت اینترنتی وجود دارند که ممکن است بخواهیم دسترسی به آن بخش ها را محدود کنیم . در یک CMSحداقل این است که یک بخش مدیریت (admin) وجود دارد که ویرایشگر سایت از طریق این بخش امکان تغییر در محتوا و افزودن و حذف محتوای سایت را دارد . ممکن است بر اساس نیاز ، احتیاج به بخش هایی داشته باشیم که فقط شرکت ها و کاربران خاصی می بایست به آن ها دسترسی داشته باشند .
- **هماهنگی صفحات (template) :** سیستم CMS باید ظاهر و باطن هماهنگ و یکنواختی داشته باشد . عناصر طراحی باید از عناصر منطقی جدا شوند . مثلاً برنامه ای که برای نمایش یک مقاله مورد نیاز است باید از چگونگی ظاهر شدن مقاله روی صفحه ، جدا شود .

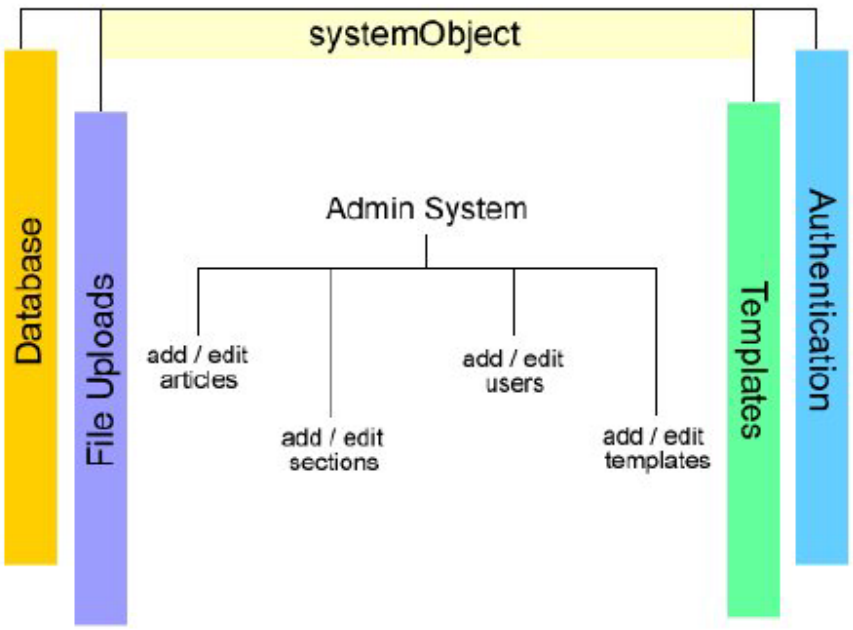
برنامه نویسی شیءگرا

PHPبا پشتیبانی از برنامه نویسی شیءگرا به فرایند طراحی کمک بیشتری می کند . هنگاهی که بخش های مختلف سیستم را کنار هم قرار می دهیم مجموعه ای از برنامه ها وجود دارند که مرتباً مورد استفاده قرار می گیرند . مانند دسترسی به پایگاه داده ها ، اعتبار سنجی کاربران و ... . برای حفظ این کدها به صورت منظم و دسته بندی شده ، ان ها را در فایل های php که کلاس نامیده می شوند نگهداری می کنیم . سپس در صورت نیاز نمونه هایی از این کلاس های را ایجاد می کنیم . به عنوان

مثال می توانیم کلاسی با کد اتصال به پایگاه داده ایجاد کنیم و سپس موقع نیاز به استفاده از query نمونه ای از این کلاس ایجاد کنیم . این روش برنامه نویسی امکان شکستن یک سیتم به مجموعه ای از بلاک های کوچکتر و ساده تر را فراهم می کند که نهایتاً منجر به مدیریت ، اصلاح و عیب یابی سریع تر می شود .

حال چگونگی هماهنگ شدن اجزا سیستم با یکدیگر را ملاحظه کنید . بدون شک این مسئله به ویژگی سیستم شما نیز بستگی دارد . در صفحه بعد طرح اولیه این سیستم را می بینید .

چهار کلاس اصلی در php وجود دارند که به صورت گسترده مورد استفاده قرار می گیرند . کار این کلاس ها عبارتند از دسترسی به پایگاه داده ها ، اجازه دادن به کاربر برای upload کردن فایلها به سایت ، خواندن و نوشتن template ها و ورود و خروج کاربران به سایت . همه این کلاس ها از یک کلاس پدر به نام system object ایجاد می شوند .



(شکل ۱-۱)

این چهار کلاس ، از یکدیگر مستقل هستند اگر چه داده های موجود در کلاس system object را به ارث می برند . تکنیک ارث بری امکان ایجاد تغییراتی که بر این چهار کلاس تاثیر می گذارد را ایجاد می کند . در وسط نمودار یک ناحیه اصلی به نام admin وجود دارد . هر بخش این سیستم نیاز به یک یا چند صفحه php برای انجام وظایف مورد نیاز دارند .

CMS ما در تعدادی پوشه که به نام های زیر نمایش ذخیره می شود :

images	templates	includes	CMSadmin
تصاویری که در CSM نمایش داده می شوند	تمپلیت برای CSM	فایلهایی که باید به بغضی صحفات ضمیمه شوند .	محدوده مدیر CMS

(جدول ۲-۱)

بخش ۲

در قسمتهای قبلی ساختار کلی را تشریح کردیم.در این قسمت کدنویسی قسمتهایی اصلی آن را شروع خواهیم کرد.

حال برای ایجاد این چهار پوشه کار را از ایجاد یک کلاس php شروع می کنیم . این کلاس ریشه admin system است و تمام متغیرها و تابع هایی که در این کلاس قرار می گیرند برای کلاس های دیگر نیز قابل استفاده هستند . این کلاس پدر را system component می نامیم که همراه با توضیحات کامل در زیر آمده است .

```
<?php
class SystemComponent {

var $settings;

function getSettings() {

// System variables
$settings['siteDir'] = '/path/to/your/intranet/';

// Database variables
$settings['dbhost'] = 'hostname';
$settings['dbusername'] = 'dbuser';
$settings['dbpassword'] = 'dbpass';
$settings['dbname'] = 'mydb';

return $settings;

}
```

```
}
?>
```

یادآوری : یک کلاس مجموعه ای از کدهاست که با اجرای این مجموعه یک شیء یا نمونه ای از این کلاس را ایجاد کرده ایم . ما می توانیم هر تعداد نمونه که بخواهیم از یک کلاس ایجاد کنیم .

کد بالا که با php شروع شده است ، کلاس system component نام دارد . بین {} متغیر $setting و تابع get setting تعریف شده اند . هدف این کلاس ذخیره کردن تعدادی از مقادیر در متغیر $setting است که محتوی آدرس سرور سایت (site dir) و جزئیات مربوط به پایگاه داده می باشد . با توجه به سیستم پایگاه داده ای که مورد استفاده قرار می دهید این مقدار را به طور مناسب تغییر دهید . در نهایت return ، $setting را به کلاس یا تابعی که آن را درخواست کرده است ارسال می کند . با پیشرفت بیشتر در کار ، داده های بیشتری در این متغیر ذخیره خواهیم کرد . کد شماره یک را با نام system component.php در پوشه includes ذخیره می کنیم .

تمام اطلاعاتی که قرار است در سیستم مدیریت محتوا نمایش داده شوند باید در پایگاه داده ذخیره گردند . برای اتصال به پایگاه داده و دسترسی به داده ها یک کلاس php قابل استفاده مجدد ایجاد می کنیم . کد زیر برای اتصال به پایگاه داده mysql است .

```
<?php

// Class: DbConnector
// Purpose: Connect to a database, MySQL version

require_once 'SystemComponent.php';

class DbConnector extends SystemComponent {

var $theQuery;
var $link;

//*** Function: DbConnector, Purpose: Connect to the database ***
function DbConnector(){

// Load settings from parent class
$settings = SystemComponent::getSettings();

// Get the main settings from the array we just loaded
$host = $settings['dbhost'];
$db = $settings['dbname'];
$user = $settings['dbusername'];
$pass = $settings['dbpassword'];
```

```
// Connect to the database
$this->link = mysql_connect($host, $user, $pass);
mysql_select_db($db);
register_shutdown_function(array(&$this, 'close'));

}

//*** Function: query, Purpose: Execute a database query ***
function query($query) {

$this->theQuery = $query;
return mysql_query($query, $this->link);

}

//*** Function: fetchArray, Purpose: Get array of query results ***
function fetchArray($result) {

return mysql_fetch_array($result);

}

//*** Function: close, Purpose: Close the connection ***
function close() {

mysql_close($this->link);

}
}
?>
```

بعد از اینکه این کلاس را db connector نامگذاری کردیم از عبارت extends system component برای بیان این مطلب که تمام داده ها و توابع از کلاس system component گرفته می شوند استفاده می کنیم . اولین تابع یعنی db connector هم نام این کلاس است . یعنی هنگام بار شدن این کلاس ، به صورت خودکار اجرا می شود . این تابع در ابتدا تابع get setting را که در کلاس قبلی تعریف کردیم صدا می زند و تمام تنظیمات پایگاه داده را از این تابع می گیرد . سپس از این تنظیمات برای اتصال به پایگاه داده استفاده می کند .

در کد صحفه قبلی مدیریت خطا را در نظر نگرفتیم اما این مطلب را در آینده اضافه خواهیم کرد .
توابع مورد استفاده دیگر ، در زیر توضیح داده شده اند .

Query: اجرای یک query پایگاه داده .
Fetch array: آرایه ای را که شامل رکوردهایی بدست آمده از اجرای تابع query است ، ایجاد می کند .
Close: اتصال به پایگاه داده را می بندد .
دستور register-shut down-function در تابع db connector ما را مطمئن می کند که در صورت عدم اتصال به پایگاه داده ، عمل close به صورت خودکار صورت می گیرد .
کد شماره دو را پوشه hncludes با نام db connector.php ذخیره می کنیم . این کلاس در سیستم داخلی زیاد مورد استفاده قرار می گیرد .
مثالی در مورد نحوه ایجاد یک نمونه از db connector :
فرض کنید که پایگاه داده ما اطلاعات مربوط به یک مشتری را نگهداری می کند و ما میخواهیم نام یک مشتری را پیدا کرده و نمایش دهیم .

```
<?php

// Get the PHP file containing the DbConnector class
require_once('DbConnector.php');

// Create an instance of DbConnector
$connector = new DbConnector();

// Use the query function of DbConnector to run a database query
// (The arrow -> is used to access a function of an object)
$result = $connector->query('SELECT firstname FROM customers');

// Get the result
$row = $connector->fetchArray($result);
```

```
// Show it to the user
echo $row['firstname'];

?>
```

برای اجرای این کلاس باید کد بالا را در پوشه includes ذخیره کنیم و جدولی با نام customers در پایگاه داده مان بسازیم . اهمیت و توانایی استفاده از پایگاه داده کاملاً روشن است . ما می توانیم اطلاعات را به یک روش استاندارد ذخیره کنیم و سپس سریعاً به اطلاعات دسترسی پیدا کنیم و یا آن ها را اصلاح کنیم و تغییر دهیم .

با استفاده از تابع query در کلاس db connector می توانیم اطلاعاتی از پایگاه داده به دست می آوریم یا در آن ذخیره کنیم . با استفاده از دستور new می توانیم یک نمونه کلاس ایجاد کنیم . این مثال می تواند کارایی استفاده از کلاس ها را نیز توضیح دهد . یعنی اگر تغییری در کلاس sestem component یجاد شود تمام فرزندان این کلاس نیز تغییر خواهند کرد .

بخش ۳

۳-۱) ایجاد پایگاه داده ها

اولین جدولی که در پایگاه داده مان اضافه کنیم مقاله ها را ذخیره خواهد کرد تا در interanet نمایش داده شوند .

قابلیت به اشتراک گذاشتن اطلاعات یکی از مهم ترین کارهایی است که شبکه سایت داخلی می تواند انجام دهد و این کار را با استفاده از سیستم مدیریت محتوا به ساده ترین حالت می توان انجام داد .

توجه داشته باشید که نیازمندی های ا طلاعاتی ما مواردی را که قرار است در جدول articles ذخیره شوند مشخص می کند . به عنوان مثال id فیلدی است که شماره مقاله را نگه می دارد . تمام فیلدهای مورد نظر در زیر آورده شده اند .

فیلد	هدف	نوع
Id	شماره هر مقاله (کلید اصلی)	Integer
Title	عنوان مقاله	Var char(300)
Tag line	خلاصه کوتاهی از مقاله	Var char(600)
Section	گروهی که مقاله به آن تعلق دارد	Integer
The article	خود مقاله	Text

(جدول ۳-۱)

قبل از ایجادخود سیستم لازم ا ست که پایگاه داده را برای ذخیره اطلاعات آن ایجاد کنیم . کد زیر چگونگی ایجاد پایگاه داده با استفاده از my sql را نشان می دهد .

```
CREATE TABLE 'databasename'. 'cmsarticles' (
'ID' int(6) unsigned NOT NULL auto_increment COMMENT 'The unique ID of the article',
'title' varchar(200) NULL COMMENT 'The article title',
'tagline' varchar(255) NULL COMMENT 'Short summary of the article',
```

```
'section' int(4) NULL DEFAULT 0 COMMENT 'The section of the article',
'thearticle' text NULL COMMENT 'The article itself',
PRIMARY KEY (`ID`)
);
```

۳-۲) ایجاد ویرایشگر:

ابتدا طراحی صفحه را با استفاده از html انجام می دهیم .

(شکل ۳-۲)

قسمت Actionlv فرم را newarticle.php مقدار دهی کرده و method آن را post قرار داده و سپس آن را درپوشه cms admin ذخیره می کنیم . در حال حاضر فیلد section از نوع textbox می باشد . در ادامه کار امکان انتخاب section را ایجاد خواهیم کرد و از نوع drop-down list برای آن استفاده خواهیم کرد .
بعد از ایجاد این فرم ، کد php مرتبط با آن را می نویسیم .

```
<?php
// Get the PHP file containing the DbConnector class
require_once('../includes/DbConnector.php');

// Check whether a form has been submitted. If so, carry on
if ($HTTP_POST_VARS){
```

```
// Create an instance of DbConnector
$connector = new DbConnector();

// IMPORTANT!! ADD FORM VALIDATION CODE HERE - SEE THE NEXT
ARTICLE

// Create an SQL query (MySQL version)
$insertQuery = "INSERT INTO cmsarticles (title,tagline,section,thearticle)
VALUES (".
"'".$HTTP_POST_VARS['title']."', ".
"'".$HTTP_POST_VARS['tagline']."', ".
$HTTP_POST_VARS['section']."', ".
"'".$HTTP_POST_VARS['thearticle']."')";

// Save the form data into the database
if ($result = $connector->query($insertQuery)){

// It worked, give confirmation
echo '<center><b>Article added to the database</b></center><br>';

}else{

// It hasn't worked so stop. Better error handling code would be good here!
exit('<center>Sorry, there was an error saving to the database</center>');

}

}
?>
```

ابتدا فایلی را که شامل کلاس db connector است ، با دستور require_once وارد می کنیم .
سپس ارسال (submit) کردن اطلاعات را چک می کنیم . این کار را با مشاهده متغیر
$HTTP-post-vars که محتوی تمام داده های ارسال شده است انجام می دهیم .
حال می بایست query را ایجاد کنیم و آن را در متغیر insert query $ قرار دهیم . قبل از اجرای واقعی query باید از دستور query که قبلاً ایجاد کرده ایم استفاده کنیم . در نهایت پیغامی مبنی بر موفقیت یا عدم موفقیت این عمل به کاربر نشان داده می شود .
تا کنون امکان درج مقاله در پایگاه داده فراهم شده است . برای بازیابی اطلاعات توسط کاربران نیز باید تمهیداتی لحاظ شود که در زیر مورد بررسی قرار می گیرند .
برای نشان دادن استخراج اطلاعات از پایگاه داده امکانی را فراهم می کنیم تا کاربران با انتخاب عنوان مقالات از یک لیست ، مقالات را ببینند . نمایش پنج مقاله جدیدتر در لیستی در صفحه اول سایت می تواند به کاربران کمک بزرگی کند .

```
<b> WHAT'S NEW: </b><br>
<?php
// Require the database class
require_once('includes/DbConnector.php');

// Create an object (instance) of the DbConnector
$connector = new DbConnector();

// Execute the query to retrieve articles
$result = $connector->query('SELECT ID,title FROM cmsarticles ORDER BY ID
DESC LIMIT 0,5');

// Get an array containing the results.
// Loop for each item in that array
while ($row = $connector->fetchArray($result)){

echo '<p> <a
href=http://mail.yahoo.com/config/login?/"viewArticle.php?id='.$row['ID'].'">';
echo $row['title'];
echo '</a> </p>';

}
?>
```

کد فوق Id و Title پنج مقاله اخیر را از پایگاه داده می گیرد و با ایجاد یک حلقه در بین آن ها هر کدام را در خطوط جداگانه نشان می دهد . این کد را در فایل index.php در پوشه ریشه (cmsadmin) ذخیره می کنیم . هر تیتری که نمایش داده می شود لینک جداگانه ای به فایل view article.php دارد . ایده ما در این کار این است که view article.php? id=1 با مقاله ای با شماره ۱ متناظر باشد و view article.php? id=2 ، مقاله ای با شماره ۲ را نشان می دهد و ... . کد زیر مربوط به view article.php می باشد .

```
<?php
// Require the database class
require_once('includes/DbConnector.php');

// IMPORTANT!!! Validate the ID number. See below
```

```
// Create an object (instance) of the DbConnector
$connector = new DbConnector();

// Execute the query to retrieve the selected article
$result = $connector->query('SELECT title,thearticle FROM cmsarticles WHERE
ID = '.$HTTP_GET_VARS['id']);

// Get an array containing the resulting record
$row = $connector->fetchArray($result);

?>

Your selected article: <?php echo $row['title'];?>
<br><br>
<?php echo $row['thearticle'];?>
```

با اجرای query و استفاده از متغیر $HTTP-post-vars شماره مقاله مورد نظر را به دست می آوریم و سپس اطلاعات مورد نیاز از جمله عنوان و متن مقاله روی صفحه نمایش داده خواهند شد .
با این کار به ویرایشگران امکان افزودن اطلاعات را می دهیم .
بدلیل اینکه ورودیهای سایت فیلتر نشده اند تاکنون کدهای نوشته شده ضعف امنیتی خواهند داشت که در قسمتهای بعدی این موضوع پیگیری خواهیم کرد .

بخش ۴

۴-۱) هویت سنجی

دو دلیل اصلی برای اعتبارسنجی ورودی کاربران وجود دارد . اولین مسئله امنیت است . با ذکر یک مثال اهمیت این مسئله را توضیح می دهیم .

فرض کنید یک query ایجاد کرده ایم که پایگاه داده را با مقداری که توسط کاربر وارد شده است مورد جستجو قرار می دهد . کد این query می تواند به شکل زیر باشد :

```
$thequery = 'SELECT * FROM products WHERE name =
"'.$HTTP_POST_VARS['uservalue'].'"';
$connector->doQuery($thequery);
```

The query is compiled from a string (SELECT...), and the user's input ($HTTP_POST_VARS...), before being executed. If the user searches for "waffles," then the query executed will be:

```
SELECT * FROM products WHERE name = "waffles"
```

Query از رشته select همراه با ورودی کاربر یعنی ($HTTP-post-vars) کامپایل می شود . اگر کاربر دنبال" waffles " بگردد query به شکل زیر اجرا می شود :

Select * from products where name=”waffles"

اگر سیستم توسط یک هکر مورد دسترس قرار بگیرد و یک کد مخرب به جای کلمه مورد جستجو وارد شود ممکن است سایت ما دیگر به خوبی کار نکند و احتیاج به برنامه نویسی مجدد داشته باشد .

بهترین راه برای حفاظت سیستم در مقابل اینگونه حملات کنترل کردن ورودی های کاربر مطابق با معیار مورد نظرمان است . اگر چه کار بسیار دشواری است و احتمالاً صد درصد نیز ایمن نیست ولی تا حدودی می تواند مسائل امنیتی را حفظ و کنترل کند .

دومین دلیل برای هویت سنجی ، راحتی مدیریت خطاهاست . مثلاً اگر فرمی داشته باشیم که شماره تلفنی را درخواست می کند نباید به کاربر امکان تایپ حروف داده شود .
راه حلی که برای اعتبارسنجی در یک CMS مورد استفاده قرار می گیرد به شرح زیر است .

۴-۲) کلاس هویت سنج

ایده ما برای حفظ مسائل امنیتی ایجاد یک کلاس هویت سنج است که بتوانیم در هر زمان ورودی کاربر را کنترل کنیم . این کلاس صحیح و ایمن بودن ورودی و عدم خطای آن را چک می کند . در ابتدا یک چارچوب برای کلاس اعتبارسنج ایجاد می کنیم .

```
<?php
require_once 'SystemComponent.php';
class Validator extends SystemComponent {

    var $errors; // A variable to store a list of error messages
    ...
}
?>
```

چهار نوع اولیه داده وجود دارند که ممکن است نیاز به اعتبارسنجی داشته باشند .

- General : فقط چیزی باید تایپ شود .
- Text only : هیچ کدام از علائم . / \ " ، و ... نباید مورد استفاده قرار بگیرند .
- فقط متن قابل قبول است و فضای خالی غیر مجاز می باشد .
- آدرس های ایمیل
- اعداد
- تاریخ ها

برای هر یک از این گروه ها می بایست روشی تعریف کنیم . منظور از روش ، مجموعه کدی است که عملی را انجام می دهند و ما آن را داخل کلاس می گذاریم . نمونه ای از این فایل ها در زیر آمده است .

```
function validateNumber($theinput,$description ){

if (is_numeric($theinput)) {

return true; // The value is numeric, return true
```

```
}else{

$this->errors[] = $description; // Value not numeric! Add error description to list of errors
return false; // Return false

}

}
```

این روش بسیار ساده است . ورودی را از کاربر می گیرد و در متغیر $ the input ذخیره می کند . در صورتی که اعتبارسنجی نتیجه منفی داشته باشد پیغامی به کاربر نشان داده می شود . این تابع متغیر $ the input را چک می کند و در صورت عدد بودن ، true برمی گرداند . در صورتی که مقدار عددی نباشد پیغام خطا در متغیر $ errors ذخیره می شود . دو روش برای اعتبارسنجی گروه های ذکر شده در بالا ، به کرّات مورد استفاده قرار می گیرند . یکی از این روش ها وجود یا عدم وجود خطا را چک می کند و دیگری لیست خطاها را در صورت وجود برمی گرداند .
حال چگونه از این کلاس جدید در فرممان استفاده کنیم ؟ این کار بسیار ساده است . ما فرمی ایجاد کرده ایم و می خواهیم آدرس ایمیل را به سایت اضافه کنیم . دو textbox در فرم وجود دارند . یکی از آن ها برای آدرس ایمیل کاربر و دیگری برای نمایش حداکثر تعداد پیامهایی که کاربر می خواهد در هفته دریافت کند .

```
<?php
// Gather the data from the form, store it in variables
$userEmail = $HTTP_POST_VARS['email'];
$maxMessages = $HTTP_POST_VARS['maximum'];

// Create a validator object
require_once('includes/Validator.php');
$theValidator = new Validator();

// Validate the forms
$theValidator->validateEmail($userEmail, 'Email Address');
$theValidator->validateNumber($maxMessages, 'Maximum number of messages');

// Check whether the validator found any problems
if ($theValidator->foundErrors() ){

// The were errors, so report them to the user

echo 'There was a problem with: '.$theValidator->listErrors('<br>'); // Show the errors, with a line between each
```

}else{

// All ok, so now add the user to the mailing list

} ?>

با کنترل معتبر بودن ورودی کاربر ریسک امنیتی را کاهش می دهیم و از ورود داده های نادرست به پایگاه داده سایت ، ممانعت به عمل می آوریم و در صورتی که کاربر پر کردن بخشی از فرم را فراموش کرده باشد این مطلب را به او یادآوری می کنیم . ما می توانیم این کد را به تمام فرم های سایت داخلی ، اضافه کنیم .

۴-۳) سازماندهی کردن

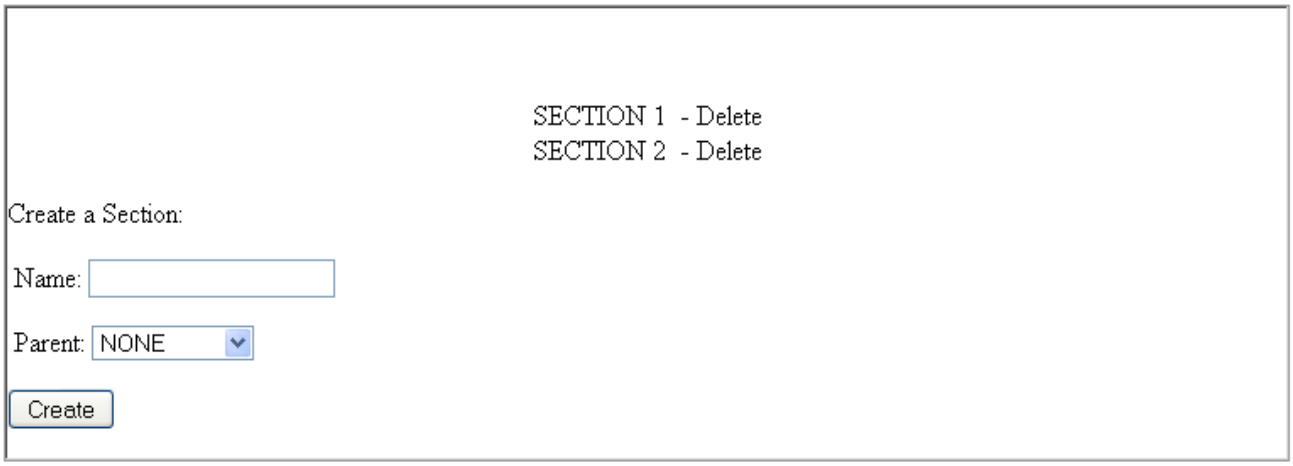
برای اینکه سایت داخلی ما با تعداد زیاد مقالات دچار بی نظمی نشود ، سیستمی برای دسته بندی مقالات ایجاد می کنیم . به این ترتیب هر مقاله یا خبر به یک دسته تعلق دارد . منظور از دسته همان فیلد Section است که قبلاً در جدول Article به آن اشاره کرده ایم .
برای اینکه بتوانیم به صورت پویا section ها را اضافه و حذف کنیم نیاز به جدول جدیدی در پایگاه داده داریم .

فیلد	هدف	نوع
Id	شماره منحصر به فرد برای هر دسته	integer
name	نام هر دسته	Varchar(20)
Parent Id	شماره پدر دسته	integer

(جدول ۴-۱)

CREATE TABLE `database`.`cmssections` (
`ID` int(4) unsigned NOT NULL auto_increment COMMENT 'The unique ID of the section',
`name` varchar(20) NULL COMMENT 'The section name',
`parentid` int(4) NULL DEFAULT 0 COMMENT 'The ID of the parent section',

```
PRIMARY KEY (`ID`)
);
```



به همان روشی که یک صفحه admin برای اضافه کردن مقالات ایجاد کردیم می توانیم صفحه ای نیز برای اضافه و کم کردن دسته ها ایجاد کنیم . ما صفحه ای به فرم زیر می خواهیم :

(شکل ۴-۳)

با باز شدن صفحه لیستی از sectionهای موجود همراه با یک لینک برای حذف آن ها (delete)نمایش داده می شود . در بخشی از این فرم می توانیم section جدید ایجاد کنیم . از یک drop_down_list برای انتخاب پدر section استفاده می کنیم . از چنین لیستی می توانیم در صفحه article برای انتخاب section ی که مقالات به آن تعلق دارند نیز استفاده کنیم .

۴-۴) نحوه عملکرد کد

با اتصال به پایگاه داده شروع کرده و یک نمونه از کلاس Validator را ایجاد کرده ایم .

```
<?php
// Require the classes
require_once('../includes/DbConnector.php');
require_once('../includes/Validator.php');
```

```
// Create an object (instance) of the DbConnector and Validator
$connector = new DbConnector();
$validator = new Validator();
```

سپس کد مرتبط با لینک deleteرا اضافه کرده ایم . توجه داشته باشید که از کلاسvalidator برای اطمینان از عددی بودن مقدار Id استفاده کرده ایم . برای تعیین عملی که باید در صفحه انجام پذیرد از متغیر action استفاده می کنیم .

http://yourinteranet/cmsadmin/sectionedit.php?action=xxx

این متغیر توسط کد زیر خوانده می شود :

```
if ($HTTP_GET_VARS['action'] == 'delete'){

// Store the ID of the section to be deleted in a variable
$sectionID = $HTTP_GET_VARS['id'];

// Validate the section ID, and if it's ok then delete the section
if ( $validator->validateNumber($sectionID,'Section ID') ){

// The validator returned true, so go ahead and delete the section
$connector->query('DELETE FROM cmssections WHERE ID = '.$sectionID);
echo 'Section Deleted <br><br>';

}else{

// The validator returned false, meaning there was a problem
echo "Couldn't delete. There was a problem with: ".$validator->listErrors();

}

}
```

کد وارد کردن section جدید بسیار ساده است . آخرین بخش کد php این صفحه لیست کردن sectionهابه همراه لینک delete کنار هر کدام از آن هاست .

```
// Execute the query to retrieve articles
$result = $connector->query('SELECT ID,name,parentid FROM cmssections');

// Get an array containing the results.
// Loop for each item in that array
while ($row = $connector->fetchArray($result)){
```

```
echo $row['name'].' -    '; // Show the name of section
echo '<a
href=http://mail.yahoo.com/config/login?/"editSections?action=http://mail.yahoo.co
m/config/login?/delete&id='.$row['ID'].'"> Delete </a>'; // Show the delete link
echo '<br>'; // Show a carriage return

}
?>
```

چیزی که کاربران مشاهده می کنند صفحه showarticles.php است که از آن برای نمایش مقالات یک گروه خاص استفاده می کنیم . مثلاً :

Showarticles.php?id="1" فقط برای اخبار یا Showarticles.php?id="2" برای مقالات جدید . عدد یک و دو نشاندهنده Id دسته (section) مربوطه هستند .

بخش ۵

مجموعه کارهایی که سیستم CMS ما باید انجام دهد :

- اطلاعات کاربران را در پایگاه داده ذخیره کند .
- کاربران را به گروه هایی مثل administrator ، editor ، staff (کارکنان) برای مشخص کردن میزان دسترسی ها طبقه بندی کند .
- امکان دسترسی فقط گروهی از کاربران را به نواحی خاصی از سیستم فراهم کند .

اولین بخشی که نیاز به ساختن آن است بخش admin می باشد که برای اضافه و حذف کردن محتوای سایت استفاده می شود . برای شروع کار جداولی را در پایگاه داده برای ذخیره کردن اطلاعاتمان ایجاد می کنیم .

Table: Groups	
Column	Purpose
ID	ID of the group
Groupname	Name of the group

(جدول۵-۱)

Table: Users	
Column	Purpose
ID	ID of the user
User	A unique username for the user
Pass	The user's password, encrypted
Thegroup	Group to which the user belongs
Firstname	The user's first name
Surname	The user's surname
Enabled	A 1 or a 0 specifies whether the user is enabled, allowing you to

	block troublesome ones

(جدول ۵-۲)

برای ایجاد جداول group از کدهای sql به شکل زیر استفاده می شود .

```
# Create groups table

CREATE TABLE `cmsgroups` (
`ID` int(4) unsigned NOT NULL auto_increment,
`groupname` varchar(15) default NULL,
PRIMARY KEY (`ID`)
) TYPE=MyISAM;

# Create 10 groups, where 1 has the highest security

INSERT INTO `cmsgroups` VALUES (1,'Admin');
INSERT INTO `cmsgroups` VALUES (2,'Editors');
INSERT INTO `cmsgroups` VALUES (3,NULL);
INSERT INTO `cmsgroups` VALUES (4,NULL);
INSERT INTO `cmsgroups` VALUES (5,NULL);
INSERT INTO `cmsgroups` VALUES (6,NULL);
INSERT INTO `cmsgroups` VALUES (7,NULL);
INSERT INTO `cmsgroups` VALUES (8,NULL);
INSERT INTO `cmsgroups` VALUES (9,NULL);
INSERT INTO `cmsgroups` VALUES (10,'Anonymous');
```

در جدول group دو فیلد وجود دارد که یکی از آن ها id می باشد . دادن مقدار یک به این فیلد نشان دهنده بالاترین امنیت است .

این دستورات را با کمک برنامه php my admin اجرا می کنیم .

برای ایجاد جدول users از کدهای sql به فرم زیر استفاده می شود و سپس با یک دستور insert ، user admin را تعریف می کنیم .

```
# Create user table

CREATE TABLE `cmsusers` (
`ID` int(4) unsigned NOT NULL auto_increment,
`user` varchar(20) default NULL,
`pass` varchar(20) default NULL,
`thegroup` int(4) default '10',
`firstname` varchar(20) default NULL,
`surname` varchar(20) default NULL,
`enabled` int(1) default '1',
```

```
PRIMARY KEY (`ID`)
) TYPE=MyISAM;

# Create sample user

INSERT INTO `cmsusers` VALUES
(1,'admin',PASSWORD('admin'),1,'Mr','Admin',1);
```

۵-۱) امنیت سایت

حال این سوال مطرح است که برای امنیت سایتمان چه کاری باید انجام دهیم . ما کلاسی به نام sentry برای کنترل log in کردن کاربر تعریف می کنیم . سیستمی که ما بر اساس آن عمل می کنیم session نام دارد ، روشی که اطلاعات کاربران را در طول دوره مشاهده سایت ذخیره می کند . تابع سازنده یعنی تابعی که موقع ایجاد کلاس اجرا می شود به فرم زیر است :

```
function sentry(){

session_start();
header("Cache-control: private");

}
```

این تابع بع php می گوید که یک session (جلسه) را آغاز کرده و یک header اضافه کند که این کار باعث حذف پسورد ذخیره شده در cache کاربر می شود .
تابع log out هم به همین سادگی است .

```
function logout(){

unset($this->userdata);
session_destroy();
exit();

}
```

این تابع متغیرهایی که شامل جزئیات کاربران ، یعنی داده session خستند را تخریب می کند و از اجرا شدن کدهای اضافی جلوگیری می کند . حال یک تابع برای چک کردن اینکه آیا کاربر قبلاً log in شده یا نه ، به وجود می آوریم و یک log in به صورت انتخابی ایجاد می کنیم . این تابع شامل پارامترهایی است .

```
function checkLogin($user = '',$pass = '',$group = 10,$goodRedirect = '',$badRedirect = ''){
...
}
```

Username و Password تابع chek log in را به سیستم می دهیم . اگر یکی از این دو (و یا هر دو) اشتباه باشند صفحه جاری آن را به آدرس ذخیره شده در badredirect $ باز می گرداند و اگر هر دو درست باشند آن ها را به goodredirect $ ارسال می کند .
group $ برای تخصیص دادن پائین ترین سطح گروه که اجازه دسترسی به منابع را دارند به کار می رود و این تابع برای ۱۰ سطح دسترسی امنیتی تخصیص داده شده که گروه ۱ بیشترین سطح دسترسی را دارد .
اگر تابع checklogin تشخیص بدهد که کاربر قبلاً log in شده باید تعیین کند که username و password قبلی هنوز معتبر است یا خیر .
اولین قسمت این تابع با بررسی اینکه آیا متغیرهای session قبلاً وجود داشته اند یا خیر ، کاربری را که به نظر می رسد log in است بررسی می کند .

```
// User is already logged in, check credentials
if ($_SESSION['user'] && $_SESSION['pass']){

// Validate session data
...

// Look up the user in the database by performing and SQL query
$getUser = $loginConnector->query("SELECT * FROM cmsusers WHERE user = '".$_SESSION['user']."' AND pass = '".$_SESSION['pass']."' AND thegroup <= ".$group.' AND enabled = 1');

// Redirect to goodRedirect or badRedirect appropriately
...
```

توجه داشته باشید که در کد بالا از تابع password در sql query استفاده می کنیم . وقتی که ما از ابتدا یک رکورد برای کاربر در پایگاه داده ایجاد می کنیم ، پسوردی ذخیره نمی کنیم در عوض از تابع password برای پنهان سازی آن استفاده می کنیم . چیزی که در پایگاه داده ذخیره می شود رشته

ای تصادفی از حروف و اعدا است و پسورد اصلی نمی تواند بازیابی شود . وقتی که می خواهیم یک log in را چک کنیم ، مانند قبل یک تابع به هم ریخته روی password ایجاد می کنیم و نتیجه را با آن رشته حروف و اعداد قبلی مقایسه می کنیم اگر مشابه باشند ، پسوردها را یکی میکنیم و کاربر قابل تشخیص است . قطع کد بعدی وقتی استفاده می شود که کاربر قبلاً یک log in نداشته است .

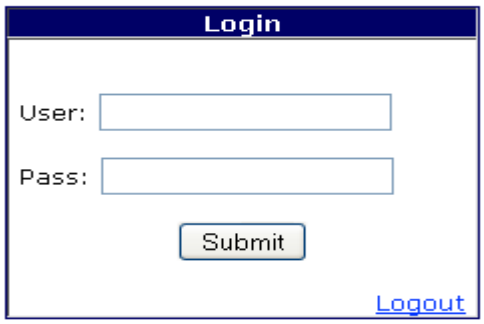
```
}else{

// Validate the input

...

// Lookup the user in the DB

$getUser = $loginConnector->query("SELECT * FROM cmsusers WHERE user =
'$user' AND pass = PASSWORD('$pass') AND thegroup <= $group AND enabled =
1");
$this->userdata = $loginConnector->fetchArray($getUser);

if ($loginConnector->getNumRows($getUser) > 0){

// Login OK, store session details

$_SESSION["user"] = $user;
$_SESSION["pass"] = $rowUser["pass"];
$_SESSION["group"] = $rowUser["thegroup"];

// Redirect if goodRedirect was provided
...

} else {

// Login BAD, Destroy session data
unset($this->userdata);

// Redirect to badRedirect if appropriate
return false;

}

}
```

اگر بیش از یک نتیجه در پایگاه داده پیدا کردیم ، یعنی جزئیات کاربردرست بوده است .

۵-۲) ایجاد فرم Log in

این فرم را به این دلیل می سازیم که به کاربری که وارد سایت شده است اجازه استفاده از کلاس sentry را بدهیم . این فرم را در هر ویرایشگر Html می توان ایجاد کرد .

(شکل ۵-۱)



باید مطمئن باشیم که عملیات فرم با صفحه ای که شامل اسم فایل است مطابق باشد ، یعنی login.php و متد آن post باشد .

```
<?php
require_once("../includes/Sentry.php");

$sentry = new Sentry(); // Create a sentry object

// Check the user's submitted login is valid
if ($HTTP_POST_VARS['user'] != ''){

$sentry-
>checkLogin($HTTP_POST_VARS['user'],$HTTP_POST_VARS['pass'],10,'welco
me.php','failed.php');

}
```

```
// Log out the user
if ($HTTP_GET_VARS['action'] == 'logout'){

$sentry->logout();

}
?>
```

آخرین مرحله کار ما در این قسمت ، امنیت صفحه می باشد . به منظور توضیح آن باید صفحه ای با نام welcome.php ایجاد کنیم که فقط بگوید :" به ناحیه admin خوش آمدید . "
برای اینکه فقط گروه های یک و دو به این قسمت دسترسی داشته باشند از کد زیر استفاده می کنیم و آن را در ابتدای فایل قرار می دهیم .

```
<?php
require_once('../includes/Sentry.php');
$theSentry = new Sentry();
if (!$theSentry->checkLogin(2) ){ header("Location: login.php"); die(); }
?>
```

این کد را در تمام صفحاتی که نیاز به محافظت دارند قرار می دهیم .
حال کار را با ایجاد یک cms با توجه به ایجاد رابط کاربر مناسب ، ایجاد یک سیستم template و چگونگی انجام کارهایی که تا کنون گفتیم به پایان می بریم .

۵-۳) نکات رابط کاربر

ده نکته ایی که در مورد رابط کاربر مناسب باید مد نظر قرار بگیرند به شرح زیر هستند :

۱. **قابل رویت بودن وضعیت سیستم :** سیستم باید به ما بگوید که در هر زمانی چه اتفاقی در حال افتادن است و نگذارد که ما در یک سری اطلاعات درهم ریخته ، سردرگم شویم . خیلی از سایت ها از یک سیستم "breadcrumb" استفاده می کنند . بنابراین موفقیت ما در سایت به شکل زیر نمایش داده می شود .

Home -> Subcat1 -> Subcat2

۲. **هماهنگی بین سیستم و جهان واقعی :** تشابهات در محاسبه خیلی مهم هستند مثل وقتی که یک دکمه را کلیک می کنیم و یک پنجره باز می شود . به عنوان مثال وقتی که شرکتی اخیراً اطلاعات خود را تقسیم بندی کرده این کار روی طراحی سایت تاثیر می گذارد .

۳. **کنترل آزادانه کاربران :** کاربر نه تنها موقعیت خود را در سایت می داند بلکه قادر است به سرعت به اطلاعات مهم در هر جایی از سایت دسترسی داشته باشد .

۴. **هماهنگی و استانداردها :** به منظور داشتن یک هدایت شهودی در سایت برای کاربران ، سایت باید کاملاً یکدست و هماهنگ باشد . به علاوه اغلب صفحات کنترلی چندین وظیفه مشترک دارند مثل cancel ، save و ... . این عملیات باید در یک موقعیت مشابه قرار داشته باشند تا ارحت تر قابل استفاده و دسترسی باشند .

۵. **پیشگیری از خطا :** پیشگیری از درمان بهتر است . واضح است که سیستم نباید توسط ویروس ها مورد حمله قرار بگیرد . یک طراحی بد به راحتی باعث اشتباه کاربر می شود .

۶. **شناسایی بهتر از فراخوانی است :** نباید کاربر موقعی که به دنبال اطلاعات کوچکی از یک لینک به لینک دیگر می رود با آیکن های خاص گیج گردد . این کار به اندازه کاری که یک کلید alt میکند برای وقتی که با موس روی یک تصویر می رویم و توضیحی برای آن نمایان می شود ، حائز اهمیت است . این موضوع شاید خیلی واضح نباشد اما به کاربران تازه کار خیلی کمک می کند .

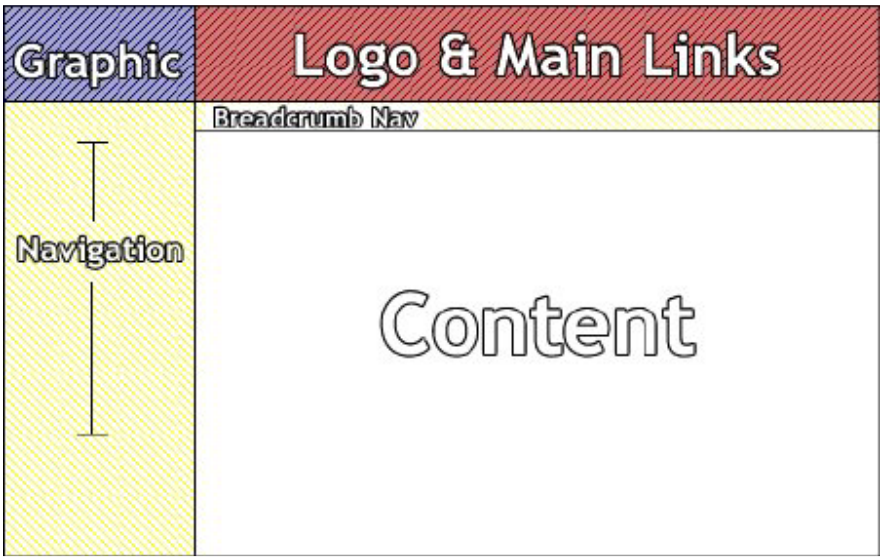
۷. **انعطاف پذیری و سودمندی :** برای اینکه سایت ما ، هم برای کاربران خبره و هم برای تازه کارها قابل استفاده باشد می بایست لینک های اساسی و پراستفاده را در سایت قرار دهیم . مانند : اخبار روز ، به روز رسانی اطلاعات و ... . مرتب سازی لینک ها و دسته بندی آن ها به کاربران کمک زیادی می کند .

۸. **طراحی زیبا و مناسب :** بعضی از cms ها به این مساله (زیبایی سایت) توجه نمی کنند در حالی که ظاهر ، بسیار مهم است . رابط سایت خود را با عملکردهای نامربوط بهم ریخته نکنید . بیشتر به مطالبی که مورد نیاز کاربران است توجه کنید . ترکیبی از زیبایی و جذابیت .

۹. **کمک به کاربر در تشخیص ، اداره و بازیابی خطاها :** این کار شامل پنهان کردن اشتباهات بد سیستم می شود . نیازی نیست به کاربر توضیحات اضافی داده شود بلکه با یک پیغام " sorry" می توان اشتباه را توجیه کرد .

۱۰. **help و مستند سازی :** تمام اطلاعاتی را که کاربر نیاز به دانستن آن ها دارد باید در یک سند در اختیار او قرار بگیرد . help حساس به متن ، ایده خوبی است به طوری که با کلیک روی علامت سوال کوچک صفحه ، کاربر برای دانستن اطلاعاتی که لازم دارد راهنمایی شود .

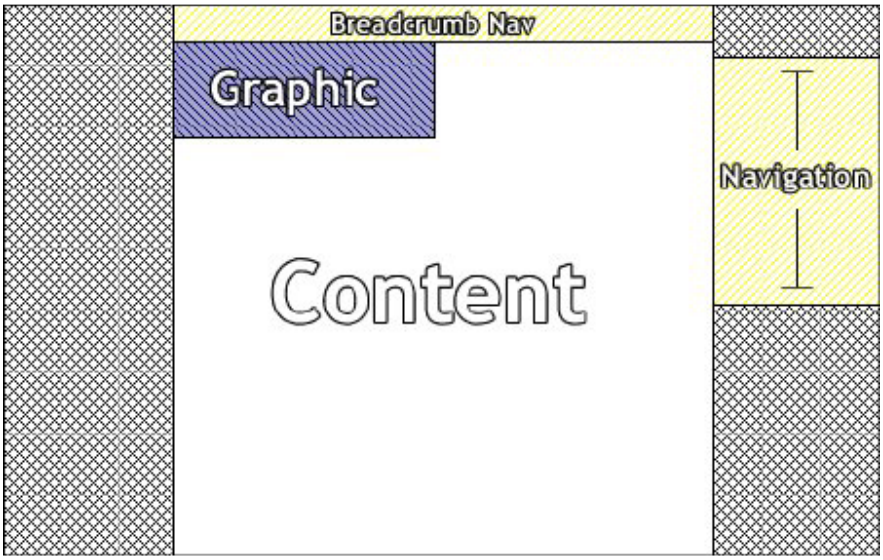
در زیر دو مدل طرح اولیه برای یک سایت همانند سایت ما آورده شده است که اولی مرسوم تر است .



(شکل ۴-۱)

در بالای صفحه(قسمت قرمز رنگ) تعدادی لینک اصلی نمایش داده شده است . مهم است که آن ها نزدیک به یک لوگوی بزرگ باشند تا کاربر بتواند به راحتی روی آنها کلیک کند . navigation (هدایت) در دو قسمت است : breadcrumbs که در بالای متن اصلی است وموقعیت لحظه به لحظه کاربر را نشان می دهد و منوی navigation در طرف چپ . همچنین کاربر به وسیله یک searchbox که در بالای این قسمت قرار دارد هدایت می شود .

یکی از عیوب این طرح این است که به اندازه کافی زیبا نیست و وقتی که شفافیت صفحه مانیتور پائین باشد مطالب اصلی به راحتی در هم ریخته می شوند .



(شکل ۴-۲)

این طرح بر روی مرکز متمرکز شده و بیشتر شبیه به روزنامه های سنتی است تا شبیه به طرح بندی یک سایت . مستطیل navigation در سمت راست به کاربر اجازه می دهد که قسمت مورد نظر خود را انتخاب کند و ستون بالا نشان می دهد که کاربر الان در کجای سایت است . از معایب این طرح قسمت های خالی در طرفین متن است . این فضاهای خالی ، توجه کاربر را به اطلاعات اصلی جلب می کند اما بسیاری از فضای صفحه را هم ، هدر می دهد .

در واقع انتخاب یکی از این دو طرح بستگی زیادی به متن مورد نظر دارد .

۲-۵) سیستم های قالب بندی (Template)

ایجاد یک سیتم قالب بندی در cms ما می تواند مفید واقع شود . php ایی که در اینجا لازم است پیشرفته تر از سایر قسمت هاست اما کدهای کاملی برای download در اختیار ما قرار دارد .

وظایفی که template ها انجام می دهند عبارتند از :

- به کاربرانی که دانش فنی کمی دارند اجازه می دهد که متنی را ویرایش و یا حتی ایجاد کنند .
- اجازه تغییر صفحات متعدد template های مطلوب
- ایجاد یک ظاهر یکدست و هماهنگ در سایت
- جداسازی متن از برنامه نویسی

برای رسیدن به این هدف ، روش های متعددی وجود دارد . اصلی ترین و معمول ترین روش این است که متن ها را در فایل های جداگانه قرار دهیم و وقتی به آن ها نیاز داریم با استفاده از دستور

include ($filename) از آن ها در فایل مربوطه استفاده کنیم . این یک راه حل ایده آل نیست زیرا فایل ها با روشی نامتعارف ساخته و ایجاد می شوند .

۵-۳) هماهنگ سازی الگوها

یک راه ساده و خوب برای هماهنگ سازی الگوها این است که یک موتور هماهنگ ساز ایجاد کنیم . به عنوان مثال باید یک فایل template که شبیه کد زیر است ایجد کنیم :

```
<i> My name is {name}, and my favourite food is {food}</i>
```

و آن ها را با نام template.php ذخیره کنیم . (tel. مشخص کننده یک template است) حال می توانیم صفحه ای با متن ایجاد کنیم . به عنوان مثال susan.php که کد php آن مانند زیر است .

```
$values = array('name' => 'Susan','food' => 'chocolate');
compilePage('template.tpl',$values);
```

تابع compilepage ، template را با متن ترکیب می کند و خط زیر را می دهد :

My name is Susan, and my favourite food is chocolate

که به کاربر نشان داده می شود . با داشتن یک فایل template با صفحات محتوایی متعدد ، به هدف داشتن یک سایت هماهنگ و یکنواخت و جداسازی کد از مطالب دست پیدا می کنیم .
طراحی سایت با template های متناوب ، قابل تغییر خواهد بود .
این نکته یک راه حل کاملاً مناسب است هرچند که یکی از اهداف ما این است که کاربران با دانش فنی محدود هم بتوانند صفحاتی ایجاد کنند . برای اینکه کاربران به ویرایش کدهای php نیازی نداشته باشند صفحات محتوا را در پایگاه داده ذخیره می کنیم . این کار باعث می شود که کاربران به راحتی بتوانند متن ها را جستجو کنند و با یک روش معمولی ذخیره کنند .

```
foreach ($replaceTags as $key=>$val) {

$template = ereg_replace('{'.$key.'}',$val,$template);

}
```

این تابع برای تبدیل tag ها به کار می رود . مثل {food} و جانشینی با کلمات مخصوصی که در متغیر replacetag $ آمده است . در مثال بالا {name} با susan جانشین شده است و {food} با chocolate سپس صفحه حاصل در $template ذخیره شده است .
یکی از معایب ین روش این است که کاربران برای مشاهده صفحه در هر زمان احتیاج به کد بالا دارند که در یک سایت بزرگ باعث کاهش سرعت می شود . راه حل این مشکل این است که یک بار این کار را انجام دهیم و template های ترکیبی و صفحات متنی (محتوا) را در یک محیط php. یا html. برای مشاهده کاربر ذخیره کنیم . این کار " تولید صفحات ایستا " نامیده می شود و به مقدار قابل توجهی زمان load شدن سرور را کاهش می دهد .
سیستم template هایی که در این جا بررسی کردیم خیلی ساده است و ویژگی های آن معمولی است . شاید بهترین نوع template ، smarty باشد که خصوصیات پیشرفته و متنوعی دارد .

۵-۴) یک cms چه خصوصیات اضافه تری می تواند داشته باشد

در اینجا چند ایده برای اضافه کردن خصوصیات بیشتر به سایت cms ارائه می شود .
ویرایشگر WYSIWYG : ما از کارمندان خود انتظار نداریم HTML را یاد بگیرند . بنابراین چطور می توانند تغییراتی در سایت ایجاد کنند ؟! ویرایشگرهای رایگان متعددی وجود دارند که می توانیم روی سیستم خود اضافه کنیم ، مثلاً برای تهیه یک جعبه متن عادی با دکمه هایی مثل bold ، underline و italic .
Profiles : یک جدول profile ایجاد کرده و آن را به جدول cmsuser که قبلاً ساخته ایم پیوند می زنیم . این کار به ما این امکان را می دهد که برای هر کارمند یک شاخه ایجاد شود و خود کارمند هم از آن نگهداری و استفاده کند .
Sections : در این جا ما مقاله ها را با section دسته بندی کردیم . اگر سیستم طوری تنظیم شده باشد که به کاربرهای خاصی جازه دهد section ها را مدیریت کنند ، باید سیستمی را برای به روزرسانی محیط ها هم در نظر بگیریم .
Communications : جدا از اینکه کاربران برای خواندن سایت تشویق می شوند ، استفاده از سایت به عنوان یک وسیله ارتباطی ایده خوبی است . چه به عنوان یک محیط عمومی استفاده شود چه به عنوان یک سیستم برای پیام های شخصی ، فقط به یک جدول نیاز دارد که شامل "to" و "from" و یک ستون برای نوشتن message و یک صفحه Inbox که پیام ها را جمع آوری کند و جایی که در ستون to کاربرهایی را که اخیراً log in شده اند ، اضافه کند .
حال که به بحث مربوط به cms مقالات پرداختیم بهتر است روش هایی را نیز برای انتخاب یک cms بیان کنیم .

۴-۱۰) پنج مرحله که باید برای انتخاب cms مدنظر قرار بگیرند

- قبل از هر چیز باید مشخص کنیم که قصد ایجاد چه نوع سایتی را داریم ؟ آیا سایت ما Portal است و براساس روابط متقابل کاربر و مدیران سایت است؟ آیا سایت ما تنها یک Weblog است و شامل نوشته های روزمره می شود؟ آیا یک سایت آماری با هدف اطلاع رسانی می خواهیم؟ آیا سایت ما یک محیط برای انجام یک پروژه تیمی است ؟
- گام بعدی این است که مشخص کنید چه ویژگی ها و امکاناتی مورد نیاز ما است؟ آیا قصد نمایش اطلاعات به صورت پویا (Dynamic) را داریم ویا ایستا (Static) ؟ آیا نیاز به یک تقویم وقایع و رویدادها داریم ؟ آیا نیاز به یک بخش Upload/Download داریم ؟ آیا یک آلبوم تصاویر هم می خواهیم ؟ آیا به یک سیستم نظرسنجی و برآورد نیاز داریم ؟ آیا نیاز به یک تابع ترجمه داریم ؟ موارد بالا Modules یا Plug-ins نامیده می شوند و عموماً به صورت امکانات افزودنی(Add-ons) به هسته سیستم هستند . بیشتر سیستم های عرضه شده از Module های متعددی پشتیبانی می کنند اما ممکن است این امکانات جاری در نسخه های بعدی سیستم پشتیبانی نشوند و سازگار با تغییرات نباشند (Incompatible) ! تنها پشتیبانی از Module مورد نظر ما کافی نیست بلکه باید توجه کنیم که چطور امکانات مورد نظر ما کار می کنند ؟ آیا خواسته های ما را برآورده می کنند ؟ گروههای مختلف یک ویژگی را از راههای مختلف پیاده سازی می کنند ! نکته کلیدی در اینجا آزمایش است زیرا بهترین راه حل برای آگاهی از قابلیت های یک سیستم است.
- حالا که سیستم مبنا و Module های مورد نیاز خود را مشخص کردیم ، باید به نحوه نمایش سایت خود دقت کنیم (Presentation) . در حال حاضر بیشتر cms ها امکان استفاده از CSS و Template را فراهم می آورند که این امکان را به ما می دهند که به سرعت ظاهر سایت خود را تغییر دهیم (Site Skin) . محتویات مشابه با ظواهر مختلف می توانند یک منظره کاملا دراماتیک را ترسیم کنند ! این منظره می تواند خیلی جذاب باشد ویا برعکس . همه چیز به تصورات ما بستگی دارد ! منابع متعددی در اطراف ما وجود دارد که می تواند به ما کمک کنند . این منابع به صورت Template هایی هستند که توسط افرادی خلاق طراحی شده اند و در دسترس عموم قرار داده شده اند . ما می توانیم این قالب ها را Download کرده به سایت خود اضافه کنیم و با این کار رنگ و بوی تازه ای به سایت خود بدهیم .
- حالا که لایه نمایش سایت خود را تعریف کردیم باید روی محتویات سایت خود تمرکز کنیم . محتوا خیلی خیلی مهم است! بدون محتوا، هر چقدر هم که سایت زیبایی داشته باشیم، فایده ای ندارد! سایت ما بی فایده است، هیچ کاربری حاضر نمی شود برای بار دوم به سایت ما سر بزند و کم کم سایت ما محو می شود ! اطلاعات می تواند به فرم های مختلفی ارائه شود.

ممکن است مطالب کاربردی باشد یا مطالب طنز یا مطالب علمی . محتوای خوب باعث جذب مخاطب می شود . اگر یک سایت خبری داریم ، داشتن امکان تبادل اطلاعات با منابع دیگر از طریق RSS مهم است . بسته به نوع سایت ، امکانات محتوایی مختلفی مطرح می شود .

- حالا که تمام مراحل بالا را انجام داده ایم ، فقط یک قدم می ماند و آن نگهداری (maintenanc) و تجدید (renewal) سایت است . با گذشت زمان چه اتفاقی بر روی محتوای قدیمی می افتد ؟ آیا آنها آرشیو می شوند و یا در دسترس می مانند ؟ آیا این امکان وجود دارد که در صورت خرابی Server تمام سیستم بر روی یک محیط مجزا بازسازی مجدد (Restore) شوند ؟ آیا امکان گرفتن نسخه پشتیبان (Backup) در سیستم وجو دارد یا این کار باید به صورت دستی صورت بگیرد ؟ این ها سوالاتی هستند که با پاسخ دادن به آن ها می توانیم cms مناسبی را ایجاد کرده و یا تهیه کنیم .

نتایج و پیشنهادات

طرحی که ما در اینجا ارائه کردیم ، فقط یک نمونه می باشد و تنها اصول اولیه این کار در اینجا بیان شده است . برنامه نویس می تواند با توجه به نیازهای خود و موضوعی که انتخاب می کند ، تغییرات متناسب را بوجود آورد . مثلا در موضوع مقالات که مورد بررسی ما بود ، می توان فیلدی برای نویسنده مقاله و نیز فیلدی برای تاریخ آن اضافه کرد . اینگونه تغییرات با توجه به نیازهای سایت و کاربران آن تعیین می شود . همچنین ما در اینجا برای ظاهر و رابط کاربر سایت ، بطور عملی ، حرفی نزده ایم و این مورد را به سلیقه برنامه نویس واگذار کرده ایم .

در نهایت اینکه در اینجا تنها قدم های اولیه برای نوشتن یک cms بیان شده و قدم های بعدی بر عهده خود برنامه نویس می باشد .

منبع :
http://www.intranetjournal.com