Extracted from:

ExpressionEngine 2
A Quick-Start Guide

This PDF file contains pages extracted from *ExpressionEngine 2*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit http://www.pragprog.com.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.
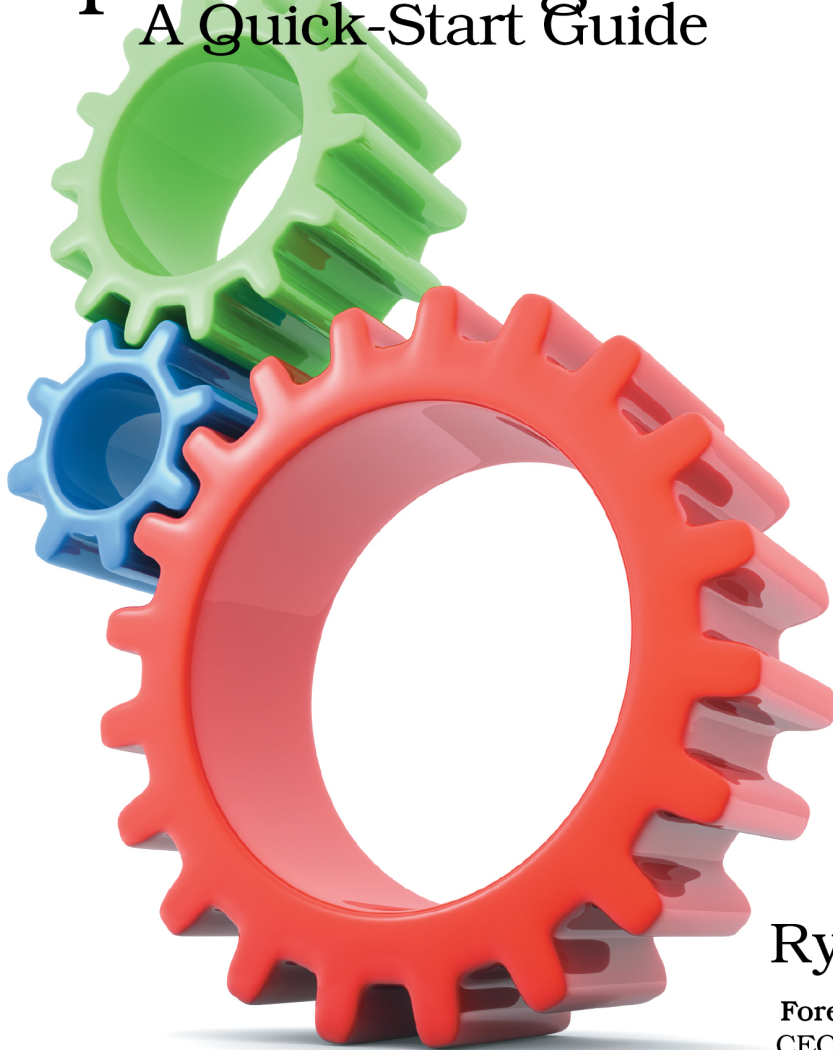
The Pragmatic Bookshelf
Dallas, Texas • Raleigh, North Carolina

ExpressionEngine 2
A Quick-Start Guide

Ryan Irelan

Foreword by Rick Ellis,
CEO of EllisLab, creator
of ExpressionEngine

Edited by Susannah Davidson Pfalzer

In the previous chapter, we set up ExpressionEngine to handle all of the content of our *Engine City Times* website. In this chapter, we'll reap the benefits of the work we've done and begin to see the *Engine City Times* website come together. We'll focus on creating the templates and on using Expression-Engine to display content, list categories, and add commenting functionality to articles.

We've made a lot of progress so far, but there is still work to do to bring the website to life.

5.1 Building Out the Home Page

Let's first change things up a bit and start off with a little DIY work for you. Before we can jump in and create our templates, it would make sense to have some content to display. We could certainly create the templates without content, but it would be a challenge knowing whether the code we're writing is actually working correctly.

Creating Placeholder Content

For the front page, we need to create four news articles. We need one article to be a Featured Top Story and the rest to be Top Story articles. Click the Content button, click Publish, and then choose News Articles. This will bring you to the news articles publish form, as shown in . Give the article a title, excerpt, and body. You can see that I'm using a fake title and some fake Latin text for the news copy. Since this is just to help us out while coding the templates, it's not important that this content makes sense or is even real. The people at *Engine City Times* will do all of the data entry, so we just have to put enough in to help us build the site.

After you entered the content, click over to the Categories tab, and choose World, which is nested under News. We want to assign all the sample content to the same category so that later in the chapter we can easily build the category template. (Since all of the category pages use the same template, once we build one, we've built them all.)

Click over to the Options tab. Here we need to set the status of the article. You'll remember from the previous chapter that we created two custom statuses for new articles: Top Story and Featured Top Story. Let's make this article the one that appears at the very top of the site (there can be only one) by assigning it the Featured Top Story status. Click Submit to add the article and publish it on the site.

Figure 28—Adding sample content to the News Article section

On your own, create three Top Story articles using the news articles publish form. Don't forget to use the proper status!

When you've completed creating the placeholder content, you should have four articles listed when viewing the Edit screen. Three articles should have a status of Top Story and one Featured Top Story.

The last set of placeholder content we have to create is for the Letters to the Editor section. This section appears at the bottom of the home page and contains three very short letter from readers, displayed in their entirety. Navigate to the letters publish form using the Content button. Give each letter a title, author name, and letter copy. Letters don't have any categories or custom statuses, so we can fill out the form and just click Submit to save the letter and publish it to the website. With those three letters created, we should now have seven pieces of content—four news articles and three letters to the editor—listed on the Edit screen.

Starting Static

Open the directory of *Engine City Times* templates you downloaded in Chapter 3, *What We're Building,* on page ?. Find the img directory, and upload it to the root of your website on the server. This directory contains a few images we need for the site.

With that uploaded, open the template template-home.html in your favorite text editor. Using the Design button, go to the Template Manager. Copy the static template—in its entirety—and paste it into the index template under the site group in ExpressionEngine. Click Update to save the changes to the template.

Open the site in your browser (or click the gray View Rendered Template button). You'll notice that the template changes are there, but there is no styling. Why? Because we haven't added the CSS that makes the page look nice. Let's do that now.

There are two ways to manage CSS files for ExpressionEngine-powered websites. There is the traditional way of having the CSS file live somewhere on the server and just referenced in the head element of the HTML document. With ExpressionEngine you can also put your CSS in a template, which keeps it together with the rest of your template code.

We didn't create a CSS template in the previous chapter, so let's do that now. Navigate to the site template group in the Template Manager, and create a new empty template named site_css. Choose a Template Type value of CSS. Click Create to add the new template.

In the downloaded directory of site template files, open core.css, which is located in the css directory. Copy the entire contents of the file into the newly created site_css template, and save it.

There's one more step left before the styles will be applied to the index template. We need to link the stylesheet template in the index template. Open the index template, and find this line near the top:

```
EngineCityTimes/template-home.html
<link rel="stylesheet" href="css/screen/main.css"
type="text/css" media="screen" />
```

Replace the value of the href parameter with this ExpressionEngine-specific code:

```
{stylesheet=site/site_css}
```

The new CSS link should look like this:

```
<link rel="stylesheet" href="{stylesheet=site/site_css}"
type="text/css" media="screen" />
```

Reload the home page of the site in your web browser, and you should now see the page fully rendered and looking just like .

Lighting Up the Page

Now that we have the entire template in ExpressionEngine (static content and all), we can begin *lighting up*—turning the content from static to dynamic —the template using content from the ExpressionEngine database. We'll do this one small section at a time, starting with the Featured Top Story.

Bringing the Featured Top Story to Life

The Featured Top Story consists of four parts: title, author, content, and link to the full article. We'll pull all of this information out of ExpressionEngine using one tag pair:

```
{exp:channel:entries} {/exp:channel:entries}
```

Find this chunk of code in the index template that controls the display of the Featured Top Story:

```
<div id="lead-article">
 <img src="img/news-article-image.png"
  alt="Image Alt Here" />
 <h4>Duis aute irure dolor in reprehenderit</h4>
 <h5><span class="by">By</span> Joe Smith</h5>
 <p>WASHINGTON, May 20 &mdash; Duis aute irure dolor
in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit
anim id est laborum.</p>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua.</p>

<p><a href="#" class="read_story"
>Read Story</a></p>
</div><!-- END #lead-article -->
```

We'll start by wrapping all of the content in the lead-article div with a basic ExpressionEngine Channel Entries tag pair:

```
{exp:channel:entries channel=""}
```

Figure 29—Viewing the site home page with CSS included

```
{/exp:channel:entries}
```

We want to enhance it slightly using some parameters. To the tag add the limit parameter with a value of 1 because we want only one Top Featured Story to appear at a time. Also add the disable parameter, and give it a value of pagination. This tells ExpressionEngine to not make the queries necessary to create pagination since we're not using it on this page. It's a simple way to optimize your template and make it load faster.[1]

---

1. See Chapter 9, *Advanced Templating, on page ?* for more on template optimization.

You'll remember from the previous chapter that we set up ExpressionEngine to allow the newspaper editors to use statuses to determine which news articles are tagged as Top Stories and which is the Featured Top Story. To make sure we're pulling the proper content, we also have to add the status parameter to the tag pair. We want the latest entry with the status of Featured Top Story, so make that the value of the status parameter. Finally, we need to give the channel parameter the value of news_articles, which is the ExpressionEngine section from which we're pulling the content.

Your tag pair should now look like this:

```
{exp:channel:entries channel="news_articles"
limit="1" disable="pagination"
status="Featured Top Story"}
...
{/exp:channel:entries}
```

With the Channel Entries tag pair all set, let's replace the static content with ExpressionEngine variables that will pull in the content from the database. The tags we use are just the short names of the fields we created in the previous chapter. For this chunk of content, we will use {title} and {article_excerpt}. We'll populate the author byline using the {author} variable that ExpressionEngine makes available for every entry.

Place these variables in their proper places, replacing the static content in the template. That part of your template should look like this:

```
MakingthePages/template-home-ee.html
<div id="lead-article">
 {exp:channel:entries channel="news_articles"
 limit="1" disable="pagination"
 status="Featured Top Story"}
<img src="img/news-article-image.png" alt="" />
<h4>{title}</h4>
<h5><span class="by">By </span> {author}</h5>
{article_excerpt}
 <p>
      <a href="{title_permalink="news_articles/view"}"
class="read_story">Read Story</a></p>
 {/exp:channel:entries}
 </div><!-- END #lead-article -->
```

Besides the article image, which we'll ignore for now and come back to later, the only thing left to light up is the Read Story link that leads you to the full article. There are several different types of links you could create, but for our purposes we want the link to have the article title in the URL.

We'll use the {title_permalink} single variable and specify what template group and template the link should point to. We want to use a template that will show the entire article (more on that in Section 5.4, *Building Out the Article Template*, on page ?). Let's assume the template will be named "view." We'll want to create the *permalink*, the unique URL where this article will live, like so:

MakingthePages/template-home-ee.html
```
<p>
      <a href="{title_permalink="news_articles/view"}"
class="read_story">Read Story</a></p>
```

This variable will render a link that contains the article title at the end, preceded by the news_articles template group and view template.[2] We'll make that URL work shortly.

That's it for the Featured Top Story. Now we need to light up the rest of the top stories.

Let's review where we are. Up to this point you should have pasted in the static home page template and created a CSS template with the site CSS. Finally, you should have coded the Featured Top Story at the top of the home page of the *Engine City Times* website.

Secondary Top Stories

The secondary top stories appear in two columns below the Featured Top Story. The left column contains two stories, and the right column contains just one. This presents a slight challenge because we have to be smart about how we pull this content in from ExpressionEngine without breaking the page layout. We'll do each side separately and add a special parameter that will ensure we have unique content on both sides.

The left column contains two featured news articles, so we want to wrap it in an Channel Entries tag pair just like we did previously. The only difference this time is we set a different value for the status parameter and for the limit parameter. The status parameter value should be Top Story so we pull in the latest stories set as top stories. Set the limit parameter to 2 so we display only two stories on the left.

Use the same variables as we did on the Featured Top Story to light up the content. Since ExpressionEngine will loop through all entries based on the parameters, you need only one of the blocks of markup.

---

2. Your exact URL will look different, but here's an example: http://example.com/index.php/news_articles/view/sample_news_article/.

Your final code for displaying both top stories should look like this:

```
{exp:channel:entries channel="news_articles"
limit="2" disable="pagination" status="Top Story"}
<h4>{title}</h4>
<h5><span class="by">By </span> {author}</h5>
{article_excerpt}
<p><a href="{title_permalink="news_articles/view"}"
class="read_story">Read Story</a></p>
{/exp:channel:entries}
```

Save the changes to the template, and then view the site home page in your browser. Two articles from the ExpressionEngine database should appear in place of the static content that was there. The Read Story link should point to a URL that has the title in it. If everything looks good, let's move on to the single story that populates the right column.

The right column presents us with another challenge. If we just put in the same parameters as the left side, with the only change being limiting it to one single article, we'd see the same article on the right that is listed first on the left. So, how do we get around this?

The key is to use a parameter called offset. This parameter allows us to display an entry that is not the most recent but, for example, the latest minus 2. Add the Channel Entries tag pair to the right side, and replace the static content with the entry variables just like we did with the first two sets of content. The parameters should change, so we display only one story and then add the offset parameter and assign it a value of 2.

Your code should look like this:

```
{exp:channel:entries channel="news_articles"
disable="pagination" limit="1"
status="Top Story" offset="2"}
<h4>{title}</h4>
<h5><span class="by">By </span> {author}</h5>
{article_excerpt}
<p><a href="{title_permalink="news_articles/view"}"
class="read_story">Read Story</a></p>
{/exp:channel:entries}
```

Save the updated template, and then reload the site home page to view your changes. You should see all three top stories in their proper place—two on the left and one on the right. For an example of what your home page should look like, see Figure 30, *Viewing the Top Stories on the home page*, on page 13.

Figure 30—Viewing the Top Stories on the home page

Letters to the Editor

The last part of the home page content that we need to build out is the Letters to the Editor section at the very bottom of the page. This section contains three short letters from the newspaper readers. It includes a title, an author, and the letter copy. The entire letter is published on the front page, so there's no link to read more.

To display the letters content, we'll use the same Channel Entries tag pair as before but give the `channel` parameter the value of `letters_to_editor` so that we're pulling content from the Letters to Editor channel in ExpressionEngine. Since there are three letters displayed at one time, we want to use a value of 3 for the `limit` parameter. This will display the three most recently added letters. The Channel Entries tag pair should now look like this:

```
{exp:channel:entries
channel="letters_to_editor" limit="3"}
...
{/exp:channel:entries}
```

Looking at the HTML in the template, we notice that each letter is contained in its own div with a unique id. These ids reference CSS, which controls how the letters are presented on the page—in three columns, one letter per column. This presents a small challenge when dynamically pulling in content from ExpressionEngine because it wants to just pull it all in without regard for layout. Our goal is to use one entry tag pair to pull in all three letters. To accomplish this, we need to use an ExpressionEngine variable called switch. This variable rotates through different values as ExpressionEngine displays each entry on the page. We want to place the switch variable in the id parameter of the div that contains the letter and give it the values of the ids, separated by a pipe (|) character.

For this example, the switch variable will look like this:

```
{switch="lcol|mcol|rcol"}
```

This tells ExpressionEngine to rotate through each of the values—in order—for each entry that it displays. Since we're displaying only three entries, it'll use each value once. If we were displaying six entries, it would use each value twice (allowing us to create two rows of three columns).

The final code in your template for the Letters to Editor channel should look like this:

MakingthePages/template-home-ee.html
```
{exp:channel:entries channel="letters_to_editor"
limit="3"}
<div id="{switch="lcol|mcol|rcol"}">
 <h4>{title}</h4>
 <h5><span class="by">From</span> {author_name}</h5>
 {letter_copy}
</div>
{/exp:channel:entries}
```

We only need one of the markup blocks because ExpressionEngine will loop through and create all three for us. Save the template, and reload the home page to see your changes. It should look similar to the screenshot in Figure 31, *Dynamically displaying the letters to the editor using ExpressionEngine,* on page 15.

With the major content pieces now being pulled from the database using ExpressionEngine, let's switch to the category list on the far-right column of the page and learn how to make a list of categories to display.
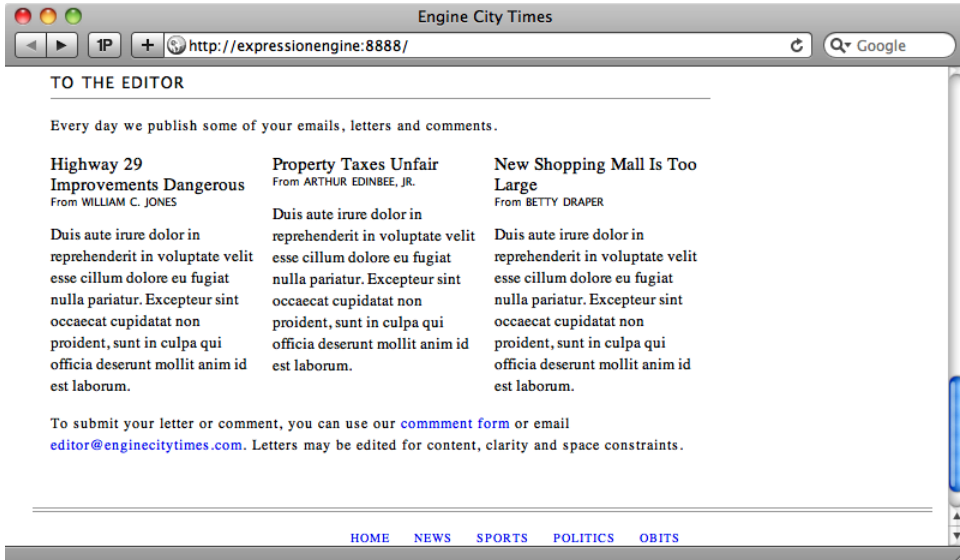
Figure 31—Dynamically displaying the letters to the editor using ExpressionEngine