Building a site with Symphony CMS

by Jonas Downey
http://jonas.do • @jonasdowney

Note: this series of blog posts was published in 2010–2011. Though some of the content is now dated, the overall principles and concepts still apply to Symphony in its current form.

YOU SHOULD USE SYMPHONY CMS, RIGHT NOW.

I'm a web content management junkie. Over the years I've used Textpattern, ExpressionEngine, Drupal, and WordPress, but none have fit my fancy as much as my recent love, Symphony.

Symphony is a beautifully minimal PHP+MySQL-based CMS that uses XML as its backbone. More on that in a minute.

More than any of the other systems I mentioned, possibly excepting Drupal, Symphony's learning curve is rather steep. For starters, Symphony's developers treat the system like a blank notebook — after installation, almost nothing is preconfigured at all. (They offer bespoke "Ensembles," which are basically sample sites you can install as a starting point.) You can build a WordPress blog in roughly 12 seconds, but making a Symphony site takes a substantial amount of forethought.

Compounding the problem is that the administrative nomenclature is somewhat obscure — what's a Data Source? An Event? A Utility? How do these relate to Pages and Sections? Fortunately, the documentation is improving, and the developers are set to make some valuable interface/conceptual improvements in the upcoming version 2.1.

Furthermore, there's that whole templating/theming issue. Symphony uses XSLT to transform its self-generated XML into standard HTML that gets displayed by the browser. This is a significant difference from the template tags approach (ExpressionEngine, Textpattern) or the direct-PHP hooks approach (Drupal, WordPress) that you might be used to. XSLT is highly powerful but it's not particularly commonplace. The XSLT requirement may continue to relegate Symphony to niche status since people may be averse to learning it, but this is a bummer, because it's more standardized (it's a W3C spec) than any of the other system-specific approaches, and it's great once you learn it.

You might be thinking, OK, this sounds hard, why would I use Symphony? The answer is that, unlike any other CMS I have tried, it lives and breathes the native tongue of the semantic web: XML. Increasingly, my online life is divided into many buckets: Twitter, Flickr, Delicious, and so on. I don't own my content anymore.

Symphony takes any XML feed you've got: RSS, Atom, SOAP, custom API output, whatever — and treats it like a first class citizen. Need to grab your most recent tweets? An RSS feed from your blog? You can do it almost instantaneously using a Dynamic XML data source. The functionality is built right in. This data gets munged together with your site's standard content into one glorious stream of XML, ready to be transformed into a new-fangled website the kids are talking about.

If that's not enough, you can use the new XML Importer extension and a simple cron job to import your XML feeds as permanent, native Symphony content. This allows you to archive and store your content — such as all of your tweets — and *regain control of your stuff*. This functionality is pure gold for Lifestreamers, who would be remiss to overlook Symphony as an option to drive their sites.

To be sure, it's not all a walk in the park. An initial site setup will take you a while. You might have to download stuff from Github. You have to learn XSLT. You will need to try some bleeding-edge software. And yes, there are ways to accomplish these tasks in other systems. But none is as conceptually elegant as Symphony. (For example, importing generic XML feeds from different sources in WordPress is still remarkably ugly, requiring several plugins and some custom code.)

Symphony's community is small and extraordinarily helpful, so you will get help when you need it. **If you have a lot of data and you want it back, give Symphony a try.**

I'm planning to follow this post with an overview of how I built my new site with Symphony. Stay tuned.

8 COMMENTS

Robert Stanford wrote:
Jul 6, 2010 at 2:56 am
(Edit)

Good post. I'm very fond of Symphony's minimalist approach – nothing is predefined or preconfigured, instead it makes it very easy to define your own data models, define queries to get at that data (or external feeds, as you mentioned) and use standards-based templates to transform it to HTML (or indeed any other language). It's almost more like a GUI to a simple web-application framework than a CMS. All of which makes it really flexible. Bonus.

Building a site with Symphony CMS, Part 1 | RocketFoo wrote:
Jul 10, 2010 at 12:41 pm
(Edit)

[...] As promised, I'm going to take you on a walkthrough of building a new site with Symphony CMS. I'll be referring to my new personal site to demonstrate some functional examples. If you're not sure why you should use Symphony, have a look at my previous post and then check out the Symphony beginner's guide. [...]

Guille wrote:
Sep 2, 2010 at 10:57 am
(Edit)

Well,… you've convince me

Like many others, I've also try "the Big three" (Drupal, Joomla & WP) and ended with fourth – Textpattern. The most elegant and minimal of them all, + perfect for designer (non-developer) like me.

I remember some time ago I've try Symphony and was impressed with customization of sections as core feature. But, in that time there was very little extensions even for some simple things, terrible documentation, small community… plus Txp was more than enought for me.

Don't know why, but few days ago I've decide to give it a try once more.. I explore different solutions from time to time. I'm sure I'll keep with Txp on many sites, but I think I've found replacement for Drupal, which I'm using when Txp simply isn't enought.

The "big three" have became real junkyard of confusing code, bugs and crappy plugins, and with time they'll be even worst.
On the other side, Symphony seems really clean'n'simple.
In next few months, if I get time, i'll make some comparison.

Jonas wrote:
Sep 2, 2010 at 11:10 am
(Edit)

Agreed — my experience is that Drupal and Joomla provide the opposite approach — systems that give you everything plus the kitchen sink, and your job is to take out the parts you don't want. This can be great for big sites that need all those features, but for the rest of us, not so much.

Out of the four you mentioned, Textpattern is definitely the closest in philosophy to Symphony, and it's somewhat easier to learn for novices. But it doesn't have nearly the raw architectural power of Symphony. I've also found the TXP plugin scene to be rather confusing and fragmented.

Oliver wrote:
Sep 17, 2011 at 7:19 pm
(Edit)

I am glad others feel the sa,e about Symphony as I do, just getting started with it and the template system using XML is new to me.

However Thats one negative, on the positive side, including all the things said above, its the only system I have found that can (really) easily and quickly import affiliate data feeds

Dave wrote:
Dec 24, 2011 at 11:35 am
(Edit)

Thanks Jonas, you've confirmed my suspicion. I think I'll be falling in love with Symphony.

I've download, installed and just started to dig in to the concepts. It was through the the Symphony site that I ended up here!

I knew there was a reason I bought this giant O'Reilly book, XSLT by Doug Tidwell a few years ago.

Kind regards.

Jonas wrote:
Dec 24, 2011 at 7:32 pm
(Edit)

Glad to hear it, Dave! Best of luck getting started.

BUILDING A SITE WITH SYMPHONY CMS, PART I

As promised, I'm going to take you on a walkthrough of building a new site with Symphony CMS. I'll be referring to my new personal site to demonstrate some functional examples. If you're not sure why you should use Symphony, have a look at my previous post and then check out the Symphony beginner's guide.

Disclaimer: anyone who makes websites can tell you that there are usually several ways to accomplish a task. More than likely, I have not done things the best way. Some of my motivation in posting this is to possibly get some corrections from smarter people.

For the purposes of this post, let's say you want to make a portfolio website that meets the following criteria:

1. HTML5, because Zeldman told you to.
2. A basic "about me" page with some static content.
3. A simple contact form.
4. A portfolio overview page with categorization of projects, a detail view of each project, and support for one or more images and thumbnails associated with each project. You can see mine for an example of how this might work.

- *Content imported from external sources (we'll use Twitter and Flickr) and then displayed in some meaningful fashion.*

Step 2: Figure Out Where to Build It
This is rather self-explanatory, but it bears repeating that you will need a web host (either locally or via an external provider) that supports Symphony's server requirements, which are helpfully located in the footer of getsymphony.com.

Not the best choice for hosting Symphony, but it runs Mouse Stampede like a champ.

I will add that for the purposes of building a Symphony site, it's invaluable to be able to perform occasional Unix-style command-line tasks (perhaps via SSH) and run Git. For content importing, you also need to access cron on the server. Personally, I use shared hosting at Webfaction and find it delightful for both development and production sites.

Step 3: Install Symphony and ~~Get~~ Git Familiar
The Symphony Installation Tutorial has detailed instructions, but it's better to use the Git instructions at the bottom of the page instead of the .zip package installation. Using Git greatly simplifies the process of upgrading extensions, which happens fairly often.

Since I'm a bleeding-edge kind of fella, I suggest switching to the latest development version, which is 2.0.8 RC3 as of this writing. To do so, clone the repository and then do:

```
git checkout 2.0.8RC3
```

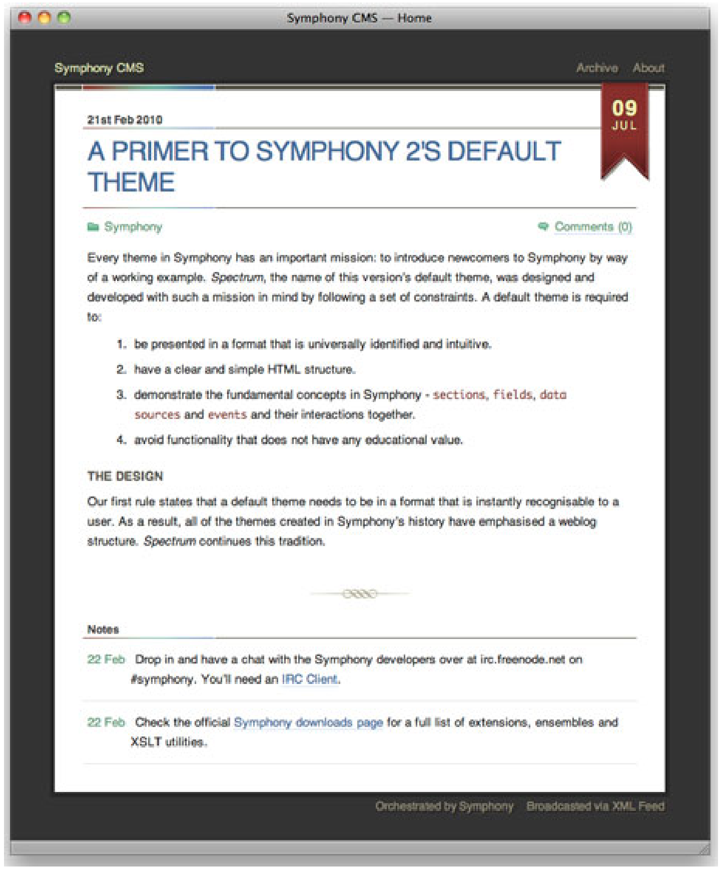
Then go ahead and install the default workspace, as it provides some nice starter material that will save us some time.

```
git clone git://github.com/symphony/workspace.git
```

If you need any help with Git, this forum thread is all you need to know.

Step 4: Say Hello to Your Little Friend
With unpleasantries out of the way, let's get going. I'm going to

assume you've worked out Steps 2-3 and installed a core Symphony installation and the default workspace. You now have something that looks like this:



21st Feb 2010

A PRIMER TO SYMPHONY 2'S DEFAULT THEME

Symphony

Comments (0)

Every theme in Symphony has an important mission: to introduce newcomers to Symphony by way of a working example. *Spectrum*, the name of this version's default theme, was designed and developed with such a mission in mind by following a set of constraints. A default theme is required to:

1. be presented in a format that is universally identified and intuitive.
2. have a clear and simple HTML structure.
3. demonstrate the fundamental concepts in Symphony - sections, fields, data sources and events and their interactions together.
4. avoid functionality that does not have any educational value.

THE DESIGN

Our first rule states that a default theme needs to be in a format that is instantly recognisable to a user. As a result, all of the themes created in Symphony's history have emphasised a weblog structure. *Spectrum* continues this tradition.

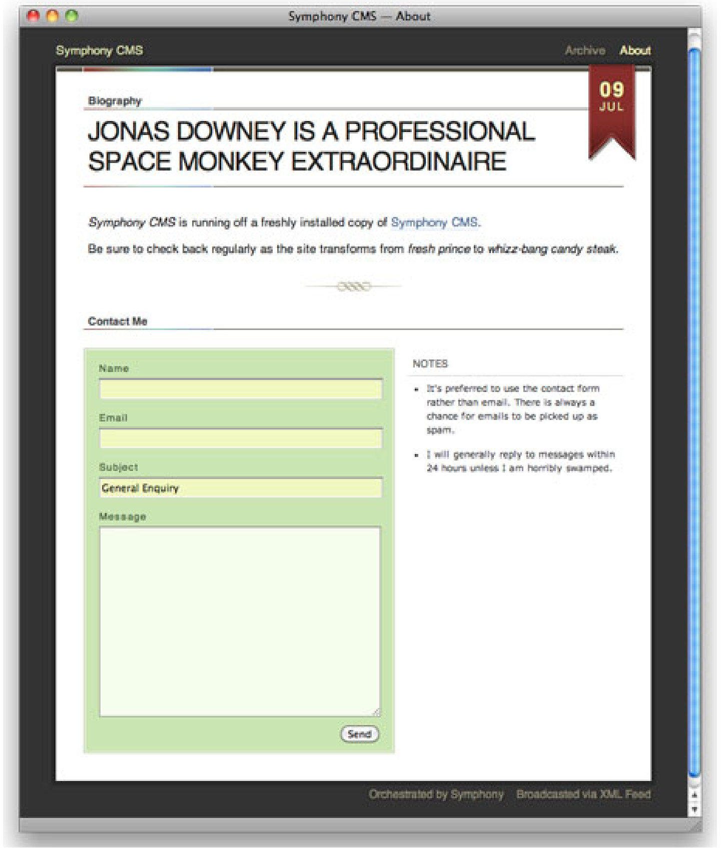
Notes

22 Feb Drop in and have a chat with the Symphony developers over at irc.freenode.net on #symphony. You'll need an IRC Client.

22 Feb Check the official Symphony downloads page for a full list of extensions, ensembles and XSLT utilities.

Orchestrated by Symphony Broadcasted via XML Feed

Good news! This default workspace already has an About page, which itself contains a working example of a contact form.



Symphony CMS — About

Symphony CMS

Archive About

09 JUL

Biography

JONAS DOWNEY IS A PROFESSIONAL SPACE MONKEY EXTRAORDINAIRE

Symphony CMS is running off a freshly installed copy of Symphony CMS.

Be sure to check back regularly as the site transforms from *fresh prince* to *whizz-bang candy steak*.

Contact Me

Name

Email

Subject

General Enquiry

Message

Send

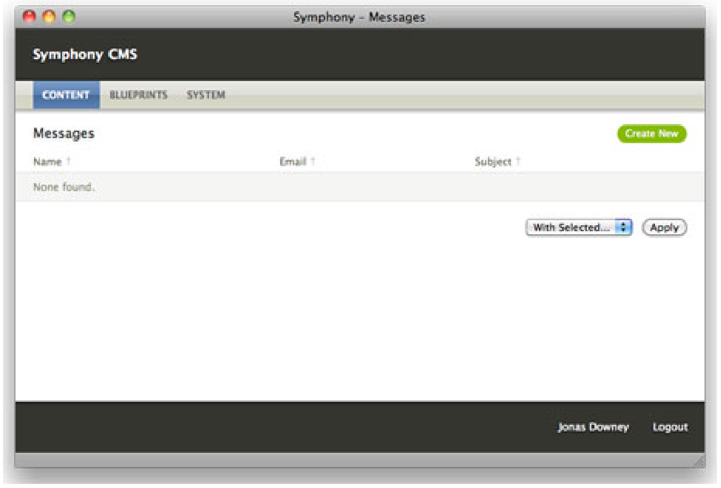
NOTES

- It's preferred to use the contact form rather than email. There is always a chance for emails to be picked up as spam.
- I will generally reply to messages within 24 hours unless I am horribly swamped.

Orchestrated by Symphony Broadcasted via XML Feed

It's also set up as a standard blog, with entries, categories, and archives. We can easily repurpose and extend this structure for projects in our portfolio, instead of blog posts.

At this point, take a moment to familiarize yourself with Symphony's administrative interface by visiting http://yoursite.com/symphony. By default, it looks like this:



Symphony – Messages

Symphony CMS

CONTENT BLUEPRINTS SYSTEM

Messages

Create New

Name	Email	Subject
None found.		

With Selected... Apply

Jonas Downey Logout

There are three main areas in the navigation. *Content* is where things like portfolio projects and blog entries will live. *Blueprints* is the gateway to the building blocks of your site — Pages, Components, and Sections (more on this later.) *System* allows you to configure various parts of your installation. I'm not going to rewrite the book on this stuff, so be sure to review the Concepts Guide.

Step 5: Pimp Your Symp

The default installation is all well and good, but it's rather sparse. Let's grab some extensions and utilities that will make things a little nicer for us going forward.

First, extensions. To make them easier to keep up to date, we'll do this Git-style using the `git submodule add` command. Here's a short list to get us started, but there are 182 of them, so use what you need! *Note: as always, run these Git commands in your site's root folder.*

XML Importer (grab external data)

```
git submodule add git://github.com/rowan-lewis/xmlimporter.git
extensions/xmlimporter
```

Codemirror (XSL syntax highlighting)

```
git submodule add git://github.com/oleae/codemirror
extensions/codemirror.git
```

CacheLite (frontend caching)

```
git submodule add git://github.com/makenosound/cachelite.git
extensions/cachelite
```

Configuration Settings (edit Symphony config from the web)

```
git submodule add git://github.com/bauhouse/configuration.git
extensions/configuration
```

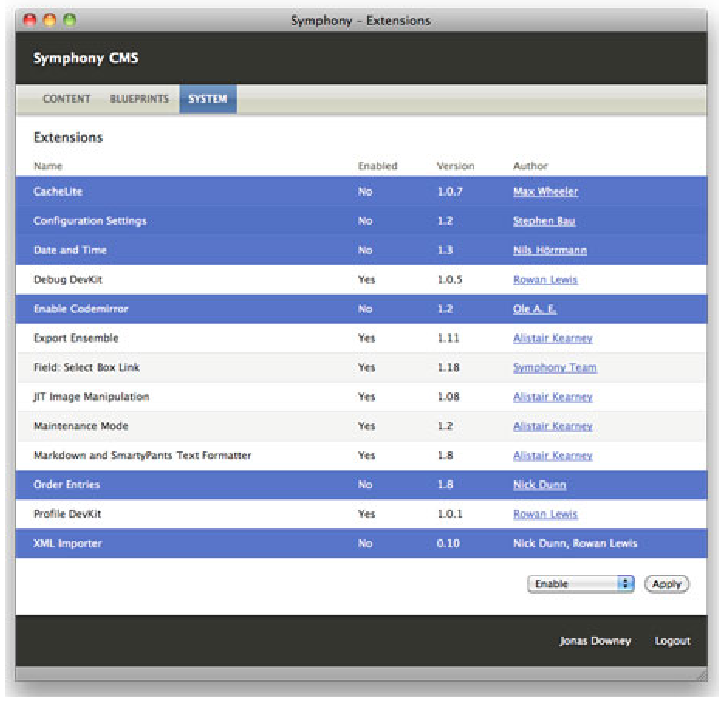
Order Entries (drag and drop content ordering)

```
git submodule add git://github.com/nickdunn/order_entries.git
extensions/order_entries
```

Date and Time (custom field)

```
git submodule add git://github.com/nilshoerrmann/datetime.git
extensions/datetime
```

Now let's enable them by going to System > Extensions. Click on the new ones and choose (With Selected) > Enable > Apply.



Extensions

Name	Enabled	Version	Author
CacheLite	No	1.0.7	Max Wheeler
Configuration Settings	No	1.2	Stephen Bau
Date and Time	No	1.3	Nils Hörrmann
Debug DevKit	Yes	1.0.5	Rowan Lewis
Enable Codemirror	No	1.2	Ole A. E.
Export Ensemble	Yes	1.11	Alistair Kearney
Field: Select Box Link	Yes	1.18	Symphony Team
JIT Image Manipulation	Yes	1.08	Alistair Kearney
Maintenance Mode	Yes	1.2	Alistair Kearney
Markdown and SmartyPants Text Formatter	Yes	1.8	Alistair Kearney
Order Entries	No	1.8	Nick Dunn
Profile DevKit	Yes	1.0.1	Rowan Lewis
XML Importer	No	0.10	Nick Dunn, Rowan Lewis

If you need to update extensions in the future, you can just run `git submodule update` on the Symphony root folder, and you're done. Woot!

Now, on to Utilities. These are little (or big) snippets of XSLT that will help us accomplish some tasks. You can add these to Symphony by copying the source code, then visiting Blueprints > Utilities > Create New. You can name each utility whatever you want, but for this tutorial, I'll be using the names in italics.

HTML5 Output
html5.xsl

Multilevel Navigation (handle sub-pages in a nav bar)
multilevel-navigation.xsl

Twita@talinkahashifyer (linkify Twitter content)
talinkahashifyer.xsl

OK, that will do for now! At this point, we've added a bit of spice to our installation and we now have the bits in place to actually get to work.

That means it's surely time to have some coffee and goof off for a while.

In part 2 of this series, I'll explain how data works in Symphony, and we'll get our content structure in place. We'll also import some stuff from Twitter. In part 3, we'll take a simple site design and convert it into Symphony pages. Stay tuned!

12 COMMENTS

Zimmen wrote:
Jul 14, 2010 at 2:48 am
(Edit)

Nice walktrough, Symphony can use more of these! looking forward to part 2.

Cena wrote:
Jul 14, 2010 at 2:28 pm
(Edit)

Nice job. Looking forward to the next one!

Brian Zerangue wrote:
Jul 15, 2010 at 2:53 pm
(Edit)

Nice tutorial!

Rowan Lewis wrote:
Jul 24, 2010 at 5:59 am
(Edit)

Did you know, Symphony actually comes bundled with a syntax highlighter as part of the debug devkit – it's in the `extensions/debugdevkit/lib/bitter` folder.

Check out the Usage section here: http://rowanlewis.com/bitter-syntax-highlighter

Jonas wrote:
Jul 26, 2010 at 6:25 am
(Edit)

Hi Rowan, does Bitter do XSLT highlighting too, or just the XML debug mode highlighting? (which I already use and enjoy, BTW)

Rowan Lewis wrote:
Jul 27, 2010 at 2:52 am
(Edit)

Yeah, it does

You can see how it handles that by debugging your master utility in Symphony 2.1.

I'm not sure how it compares, but basically it's the XML highlighter, with additional highlighting of any XPaths.

It doesn't process the XPath itself, mind you, it just colours them differently.

Rowan Lewis wrote:
Jul 27, 2010 at 2:54 am
(Edit)

Actually, that's not quite right, it seems that the version of Bitter in 2.1 is slightly out of date.

You can find full examples in the repository itself.

Building a site with Symphony CMS, Part 2 | RocketFoo wrote:
Jul 31, 2010 at 5:44 pm
(Edit)

[...] In our last episode we completed a Symphony CMS installation and loaded some nice bits of additional functionality with [...]

Timo wrote:
Feb 22, 2011 at 2:27 pm
(Edit)

Not exactly a tutorial, but surely a very nice introduction to Symphony. Thanks for that!

Leonardo A. Souza wrote:
Mar 26, 2011 at 9:51 pm
(Edit)

Good!

Building a site with Symphony CMS, Part 3 | RocketFoo wrote:
Apr 6, 2011 at 7:52 pm
(Edit)

[...] let's begin by getting our rough template in place. If you'll recall, back in part 1, we preloaded an XSLT Utility called html5.xsl (in case you forgot, it's here.) This is a [...]

Ollie wrote:

Codemirror extension needs to be

Feb 3, 2012 at 5:50 am

(Edit)

```
git submodule add git://github.com/oleae/codemirror extensions/codemirror
```

Not

```
git submodule add git://github.com/oleae/codemirror extensions/codemirror.git
```

(note no .git extension on directory)

BUILDING A SITE WITH SYMPHONY CMS, PART 2

Greetings! Last time, we completed a Symphony CMS installation and loaded some nice bits of additional functionality with Extensions and XSLT Utilities. Finally, it's time to get our hands dirty and build something.

Before we begin, please note that since our last post, Symphony has been updated to version 2.1.0. You may want to update your installation.

As you'll recall, we're going to make a portfolio site that has:

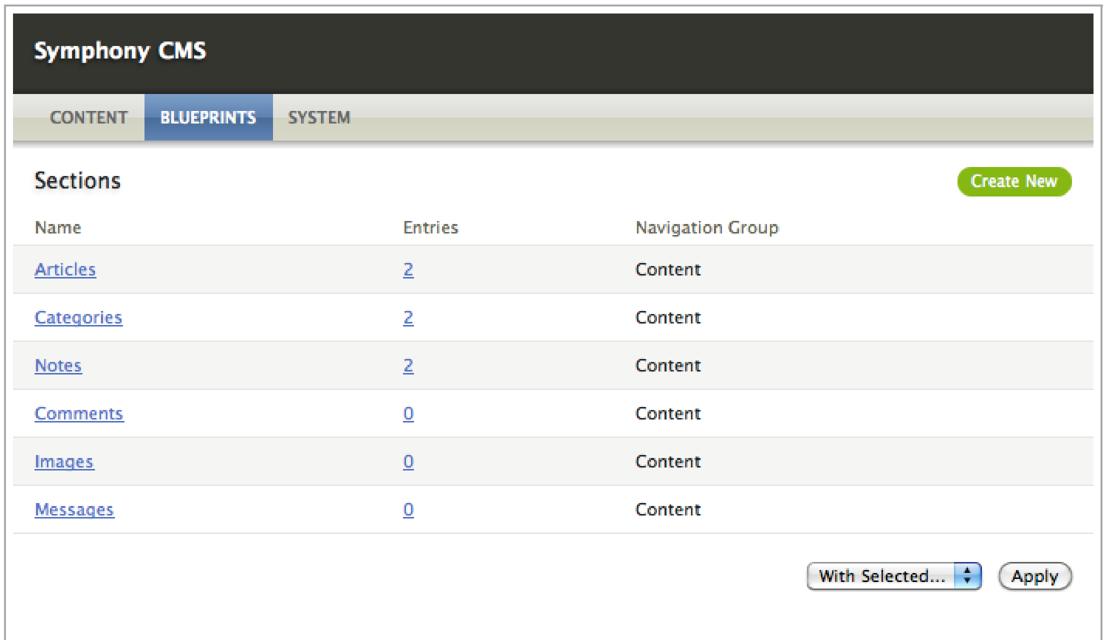
1. An overview page containing a thumbnail list of all projects organized by categories.
2. A project detail page with a big image, headline, and blurb about the project.
3. Support for storing any number of thumbnails and images per project.

OK then, let's get started. In the interest of being lazy, we're going to reuse the structure of Symphony's default workspace, which is set up as a blog, but works remarkably well for portfolio projects.

Symphony organizes content into **Sections**. You define the fields that make up a Section, using field types such as *Text Input*, *Date*, and *Checkbox*. You can also link fields to other Sections. It's powerful and awesome.

Once you have a Section defined, you will create **Entries** (i.e., blog posts, portfolio projects, or any other repetitive content such as pictures of kitties.)

Let's take a look at Sections by logging into the Symphony administration panel and selecting Blueprints > Sections.



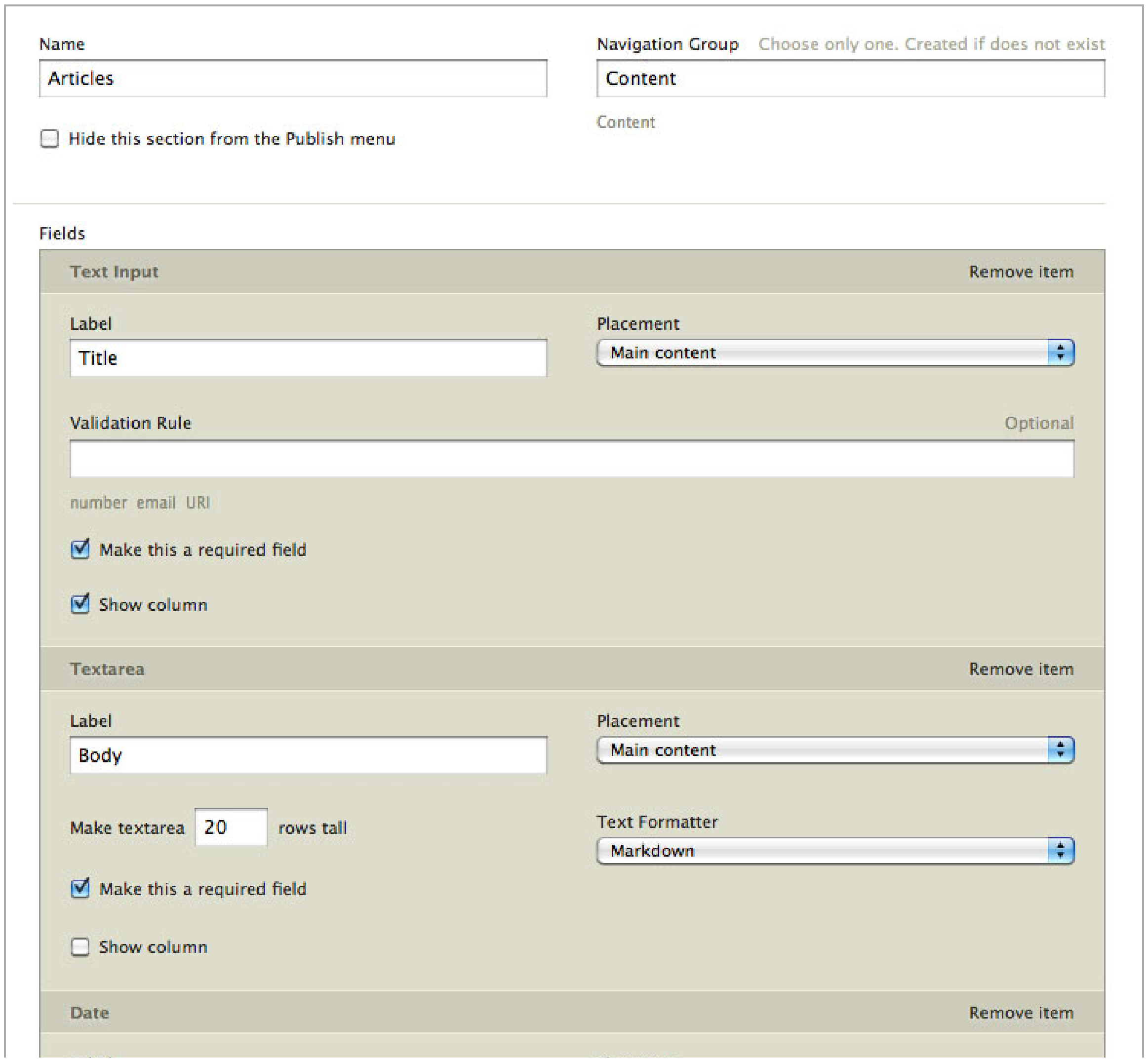
Sections

Name	Entries	Navigation Group
Articles	2	Content
Categories	2	Content
Notes	2	Content
Comments	0	Content
Images	0	Content
Messages	0	Content

Symphony's Really Excellent Default Sections™

As you can see, there are several pre-made Sections here. The ones we care about are *Articles*, *Categories*, and *Images*. You can opt to use *Comments*, too, but I'm excluding that for now.

Click on the Articles link.



Name: Articles

Navigation Group (Choose only one. Created if does not exist): Content

Content

☐ Hide this section from the Publish menu

Fields

Text Input

Label: Title

Placement: Main content

Validation Rule (Optional):

☑ Make this a required field

☑ Show column

Textarea

Label: Body

Placement: Main content

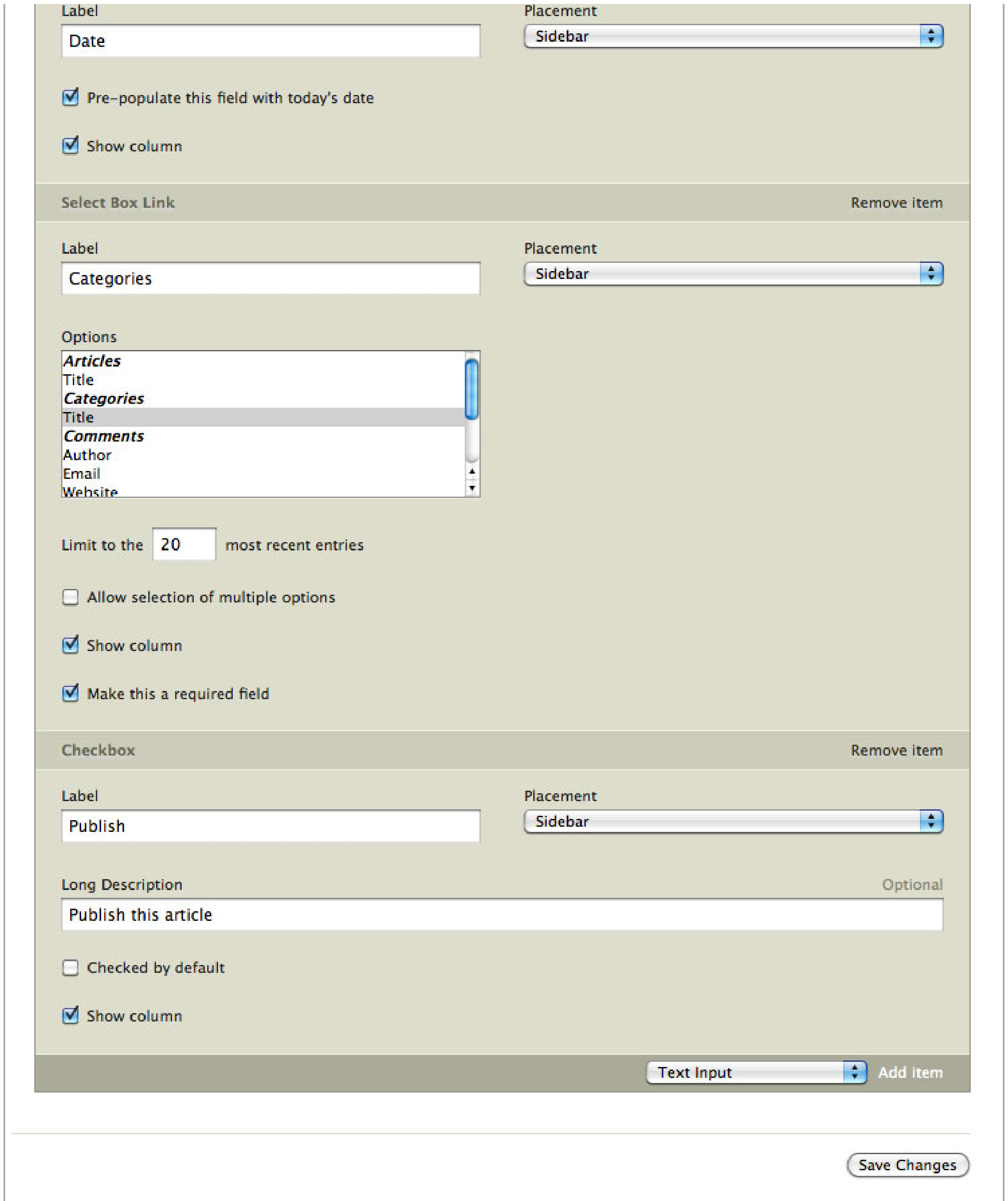
Make textarea 20 rows tall

Text Formatter: Markdown

☑ Make this a required field

☐ Show column

Date


Label: Date
Placement: Sidebar

☑ Pre-populate this field with today's date

☑ Show column

Select Box Link — Remove item

Label: Categories
Placement: Sidebar

Options:
Articles
Title
Categories
Title
Comments
Author
Email
Website

Limit to the 20 most recent entries

☐ Allow selection of multiple options

☑ Show column

☑ Make this a required field

Checkbox — Remove item

Label: Publish
Placement: Sidebar

Long Description (Optional): Publish this article

☐ Checked by default

☑ Show column

Fields in the ~~Articles~~ Projects Section

There are five fields in the Articles section: *Title, Body, Date, Categories,* and *Publish*. Conveniently, these are the essentials for our portfolio (except for Images — more on that in a minute.)

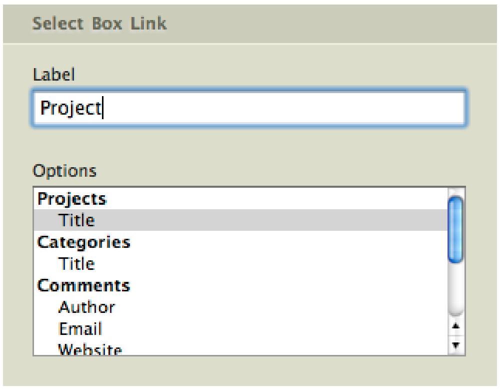
There are lots of options in the Section editor, but we can mostly ignore them at the moment — full details are in the Symphony Concepts Guide. Note that you can drag and drop a field to reorder it, and you can add fields at the bottom of the page (click on the Text Input menu to see the field types, and *Add item* to add it.)

There is only one change we need to make in the Articles section: rename the whole section to *Projects*. Do this and Save.

Now go to Blueprints > Sections and select *Images*.

Images is linked to *Projects* using a Select Box Link field. This allows us to upload as many images as we want and have them associated to a project in our portfolio. (Database geeks or people with lots of kids might call this a “one-to-many” relationship.)

This link field was originally named “Article” but we want to rename it “Project”:


Select Box Link

Label: Project

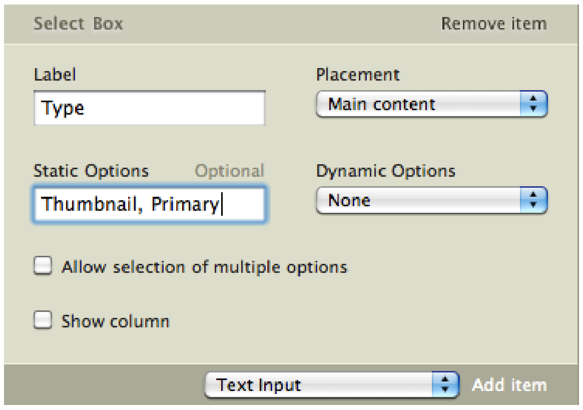
Options: Projects — Title (selected); Categories — Title; Comments — Author, Email, Website

Renaming the Select Box Link field

With that done, let’s add one more field, called *Type*. This will be a Select Box field with two options, *Thumbnail* and *Primary*, allowing us to designate which images should be used for different purposes.

Note: Symphony has an extension called Just In Time (JIT) Image Manipulation which can transform (crop or resize) images on the fly; you may prefer this for your thumbnail purposes. Personally, I’m a control freak and enjoy doing my cropping and resizing offline.

To add the field, choose *Select Box* in the Field Type menu at the bottom of the page, and click Add Item. Enter *Type* in the Label field and put the two options in the Static Options box (separated with a comma.)


Select Box — Remove item

Label: Type

Placement: Main content

Static Options (Optional): Thumbnail, Primary

Dynamic Options: None

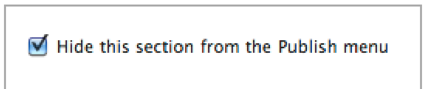
☐ Allow selection of multiple options

☐ Show column

Text Input — Add item

Typin' in Image Types

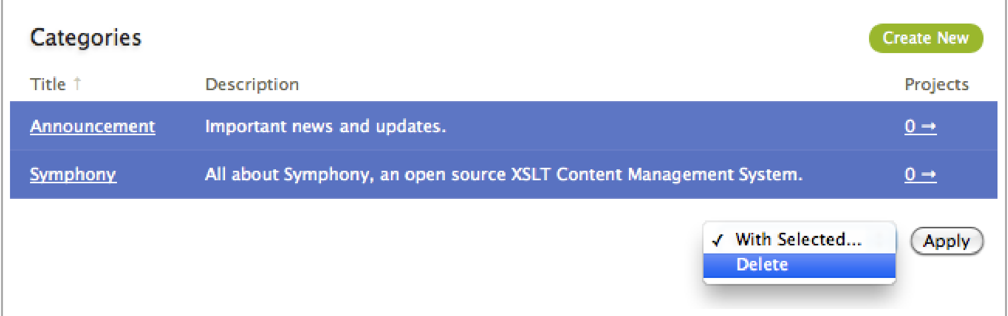
While we’re here, if you think you’ll be uploading a ton of images, you might want to “unhide” this section by unchecking this box at the top of the page:



Show yourself, Images!

now you will be able to directly select the Images section in the Content menu. Save your changes.

With the section setup done, let's look at our actual content. First, let's define some categories for our portfolio projects. Go to Content > Categories.

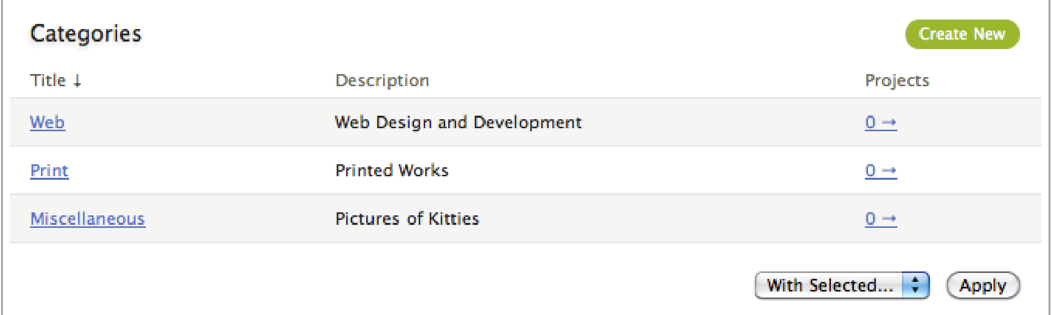


Title ↑	Description	Projects
Announcement	Important news and updates.	0 →
Symphony	All about Symphony, an open source XSLT Content Management System.	0 →

Removing the default Category entries

Hey, who put those generic Categories here? Oh right, the developers of this fine demo workspace. Despite their quality work, we don't need the sample entries, so click on each one and choose With Selected > Delete. Apply the change.

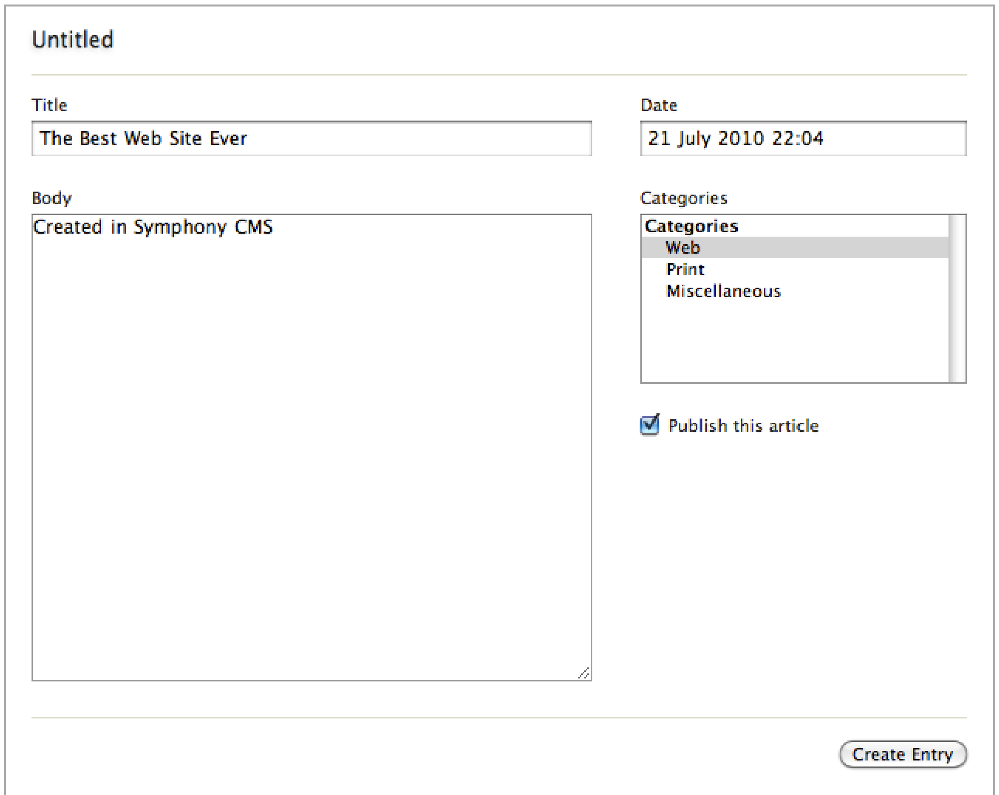
Let's make some actually useful categories. Using the **Create New** button, create categories called *Web*, *Print*, and *Miscellaneous*.



Title ↓	Description	Projects
Web	Web Design and Development	0 →
Print	Printed Works	0 →
Miscellaneous	Pictures of Kitties	0 →

Categorically Accurate

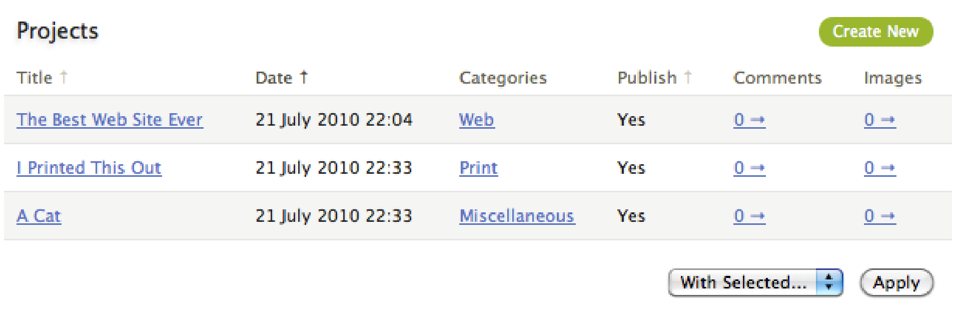
Looking good! Time to create some projects. Go to Content > Projects, and once again delete the sample items in this section, then create some dummy projects (or if you have real ones, go for it.) Be sure to add at least one project in each category.


Untitled

Title	The Best Web Site Ever
Date	21 July 2010 22:04
Body	Created in Symphony CMS
Categories	Categories: Web, Print, Miscellaneous (Web selected)
Publish this article	✓

The Best Web Site Ever

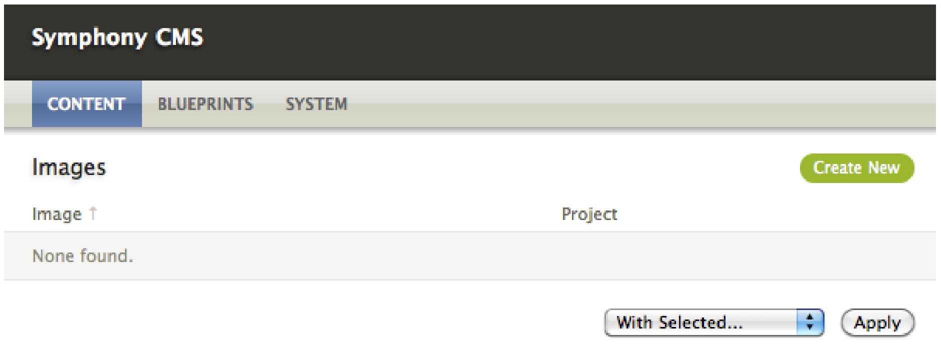
When you're done, go back to the main Projects view to take a look at your work.


Projects

Title	Date	Categories	Publish	Comments	Images
The Best Web Site Ever	21 July 2010 22:04	Web	Yes	0 →	0 →
I Printed This Out	21 July 2010 22:33	Print	Yes	0 →	0 →
A Cat	21 July 2010 22:33	Miscellaneous	Yes	0 →	0 →

Our sample portfolio. Don't expect any job offers yet.

Now we need to add some images to these projects. There are two ways to do this: either click on the "0 →" for a Project, or go to Content > Images.

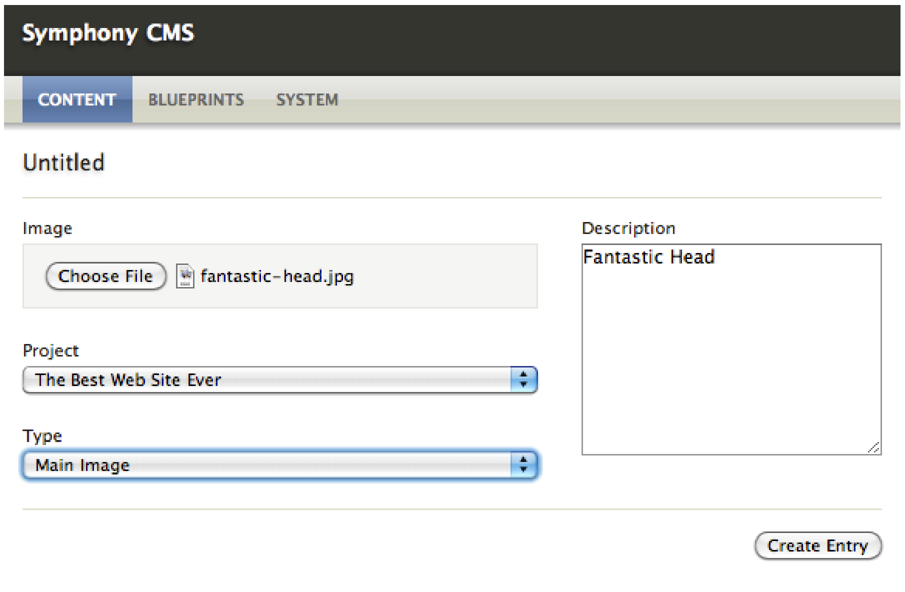


Images

Image	Project
None found.	

Gone Image Uploadin'

Click *Create New* and fill out the fields appropriately, uploading two images (Primary and Thumbnail) for each project:



Untitled

Image: fantastic-head.jpg

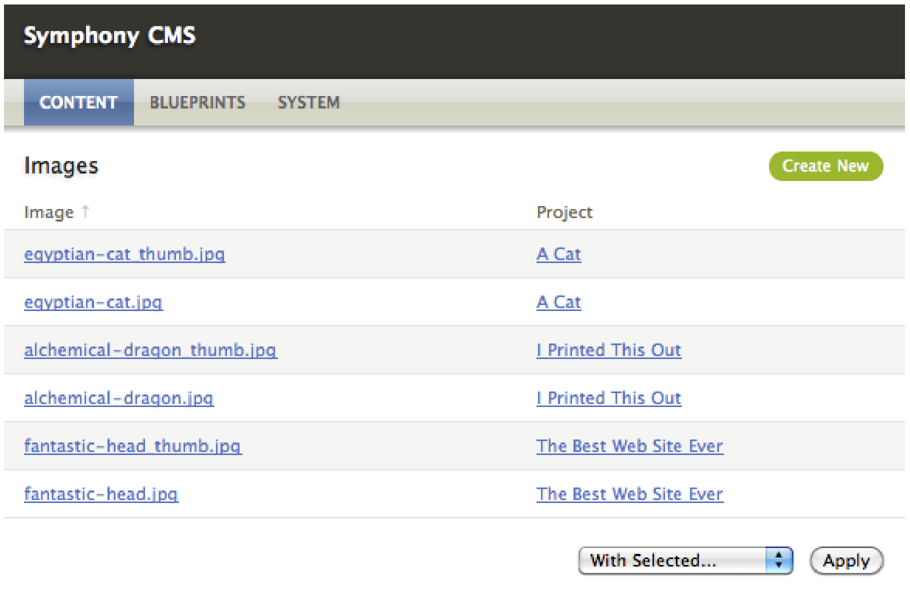
Project: The Best Web Site Ever

Type: Main Image

Description: Fantastic Head

The Best Fantastic Head Ever

When you're done, you can view all the Image entries.



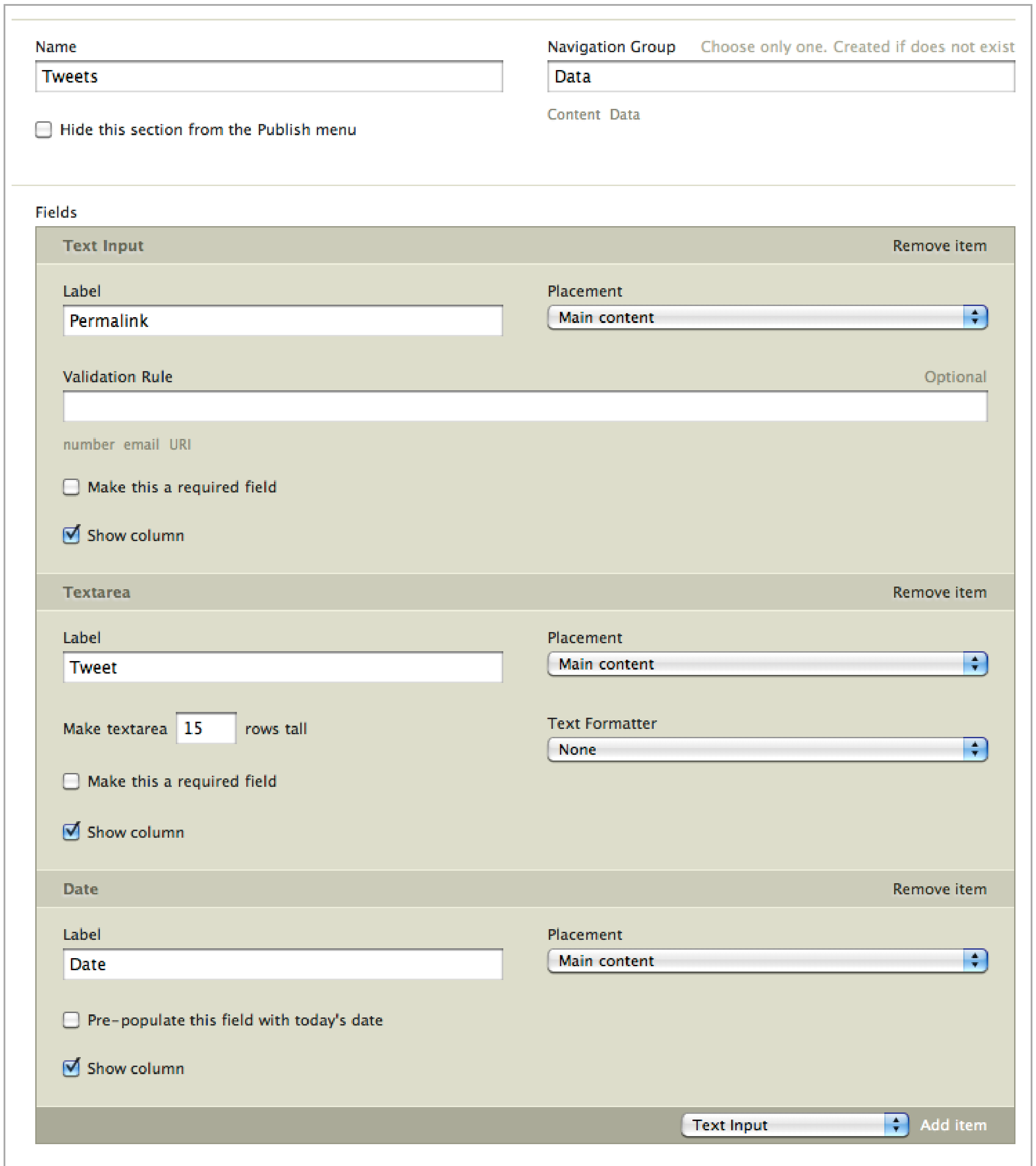
Images

Image	Project
egyptian-cat_thumb.jpg	A Cat
egyptian-cat.jpg	A Cat
alchemical-dragon_thumb.jpg	I Printed This Out
alchemical-dragon.jpg	I Printed This Out
fantastic-head_thumb.jpg	The Best Web Site Ever
fantastic-head.jpg	The Best Web Site Ever

A fleet of images

Woot. Now that we have some extremely high quality content loaded, it's time to wreck this joint. For our last trick, we're going to import some data from Twitter. Get ready, you're going to have your first taste of XPath!

First, we need to make a new section to hold our imported Tweets. I'm not going to run through this in detail, just make sure your new Tweets section matches the settings below. I like to put my imported stuff in its own Navigation Group, called "Data" – this adds a new dropdown menu in the admin interface.



Name: Tweets

Navigation Group — Choose only one. Created if does not exist: Data

Content Data

☐ Hide this section from the Publish menu

Fields

Text Input — Remove item

Label: Permalink

Placement: Main content

Validation Rule (Optional):

number email URI

☐ Make this a required field

☑ Show column

Textarea — Remove item

Label: Tweet

Placement: Main content

Make textarea 15 rows tall

Text Formatter: None

☐ Make this a required field

☑ Show column

Date — Remove item

Label: Date

Placement: Main content

☐ Pre-populate this field with today's date

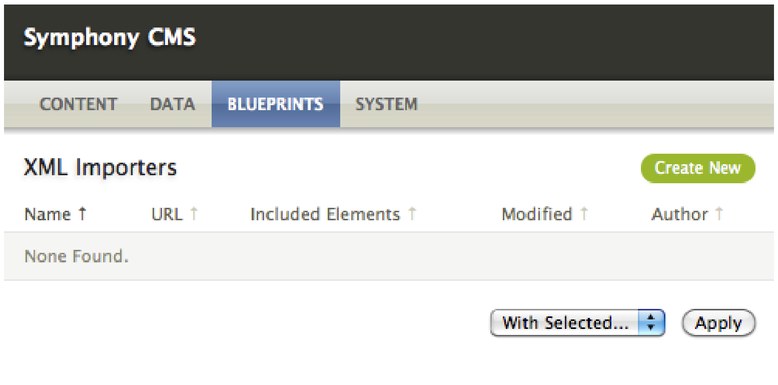
☑ Show column

Text Input — Add item

Gotta Make The Tweets

Next, we'll make an XML Importer. This does all the painful work of mapping XML from a Twitter feed to the fields in the Tweets section we just created. Once the two are hooked up, Symphony will grab Twitter's XML and put its content into your section. Each tweet becomes an Entry in Symphony, just like our projects, categories, and images.

To add the importer, go to Blueprints > XML Importer.

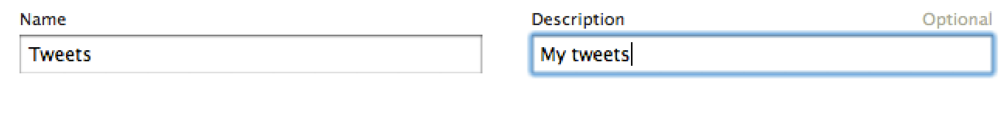


XML Importers

Name ↑	URL ↑	Included Elements ↑	Modified ↑	Author ↑
None Found.				

Bask at Your Future of XML Wizardry

Click *Create New*. Start by naming this importer "Tweets."



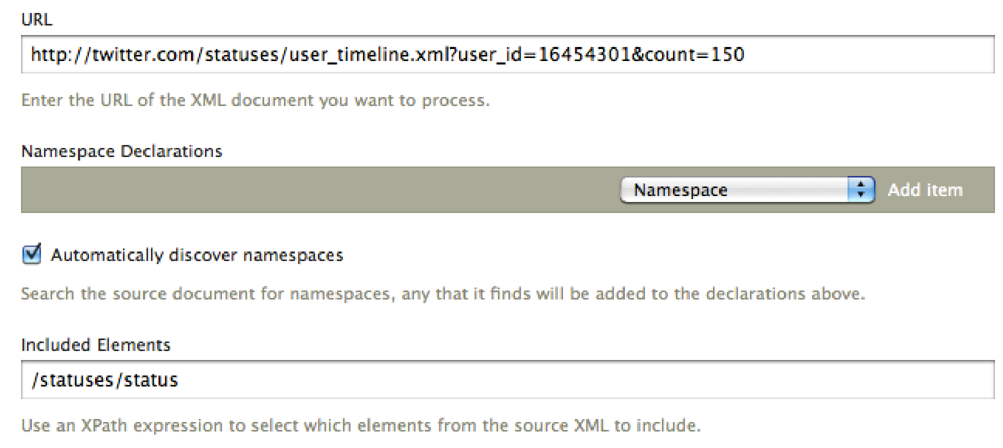
Name: Tweets
Description (Optional): My tweets

Naming the Importer

Twitter provides a bunch of different kinds of feeds via their API. I've had good success using the XML output from a URL in this format:

http://twitter.com/statuses/user_timeline.xml?user_id=16454301&count=150

You will need to swap in your own Twitter `user_id` but otherwise you can just use this as-is. Paste this URL into the **URL** field.



URL: http://twitter.com/statuses/user_timeline.xml?user_id=16454301&count=150

Enter the URL of the XML document you want to process.

Namespace Declarations

[✓] Automatically discover namespaces

Search the source document for namespaces, any that it finds will be added to the declarations above.

Included Elements: /statuses/status

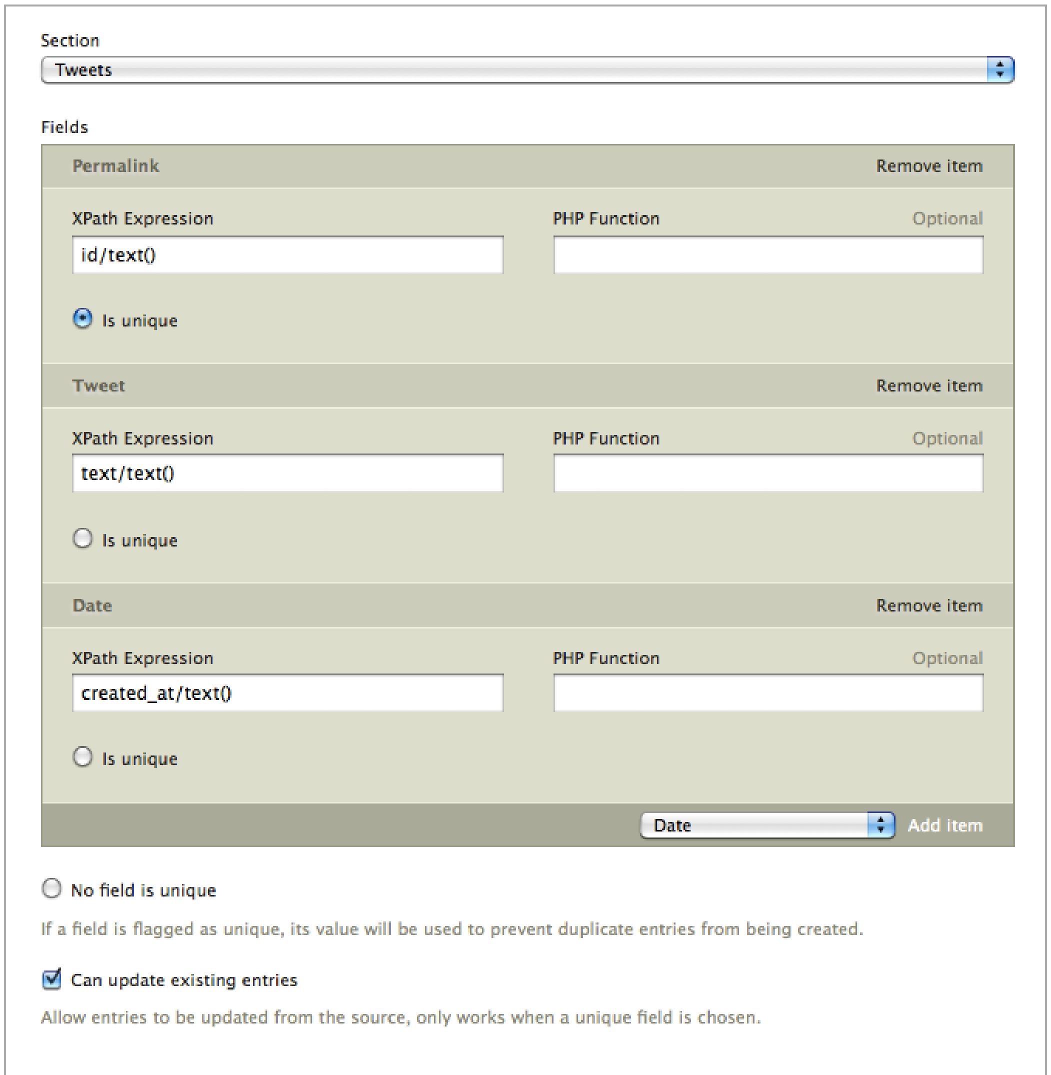
Use an XPath expression to select which elements from the source XML to include.

The faucet from which your Tweets will flow

To finish this step, check "Automatically discover namespaces." In the *Included Elements* field, enter `statuses/status`. This is a small XPath expression that selects only the `status` elements from the Twitter XML feed.

Last, we have to tell Symphony where each piece of data from a Tweet should go. In the *Destination* area, select **Tweets** from the Section field. Then add each of the three fields that make up a

Tweet (*Permalink*, *Tweet*, and *Date*.) Make sure the settings match the screenshot below (don't forget to set Permalink to "Is unique.")



Section
Tweets

Fields

Permalink — Remove item
XPath Expression: id/text()
PHP Function (Optional):
[✓ Is unique]

Tweet — Remove item
XPath Expression: text/text()
PHP Function (Optional):
Is unique

Date — Remove item
XPath Expression: created_at/text()
PHP Function (Optional):
Is unique

Date — Add item

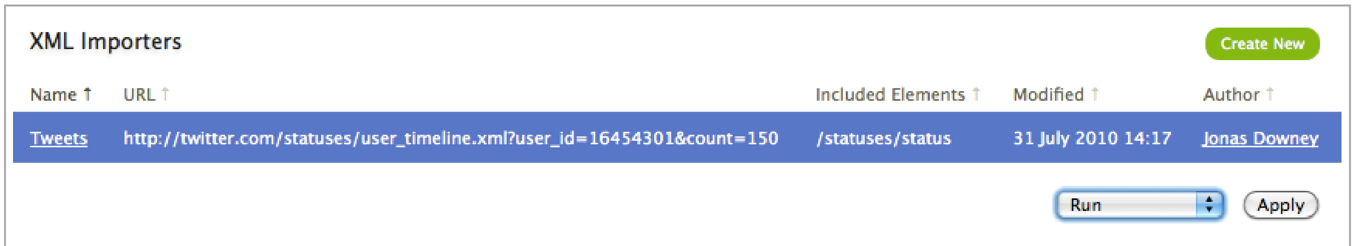
No field is unique
If a field is flagged as unique, its value will be used to prevent duplicate entries from being created.

[✓ Can update existing entries]
Allow entries to be updated from the source, only works when a unique field is chosen.

Tweet Fields

each XPath Expression selects a specific element from the Twitter XML feed and uses the `text()` function to obtain that element's content.

OK, we're done! Click **Create XML Importer**. Once it's created, go back to Blueprints > XML Importers, and click on our new Importer.



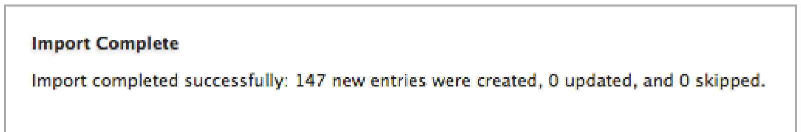
XML Importers — Create New

Name	URL	Included Elements	Modified	Author
Tweets	http://twitter.com/statuses/user_timeline.xml?user_id=16454301&count=150	/statuses/status	31 July 2010 14:17	Jonas Downey

Run — Apply

Getting Ready for a Tweet Flood

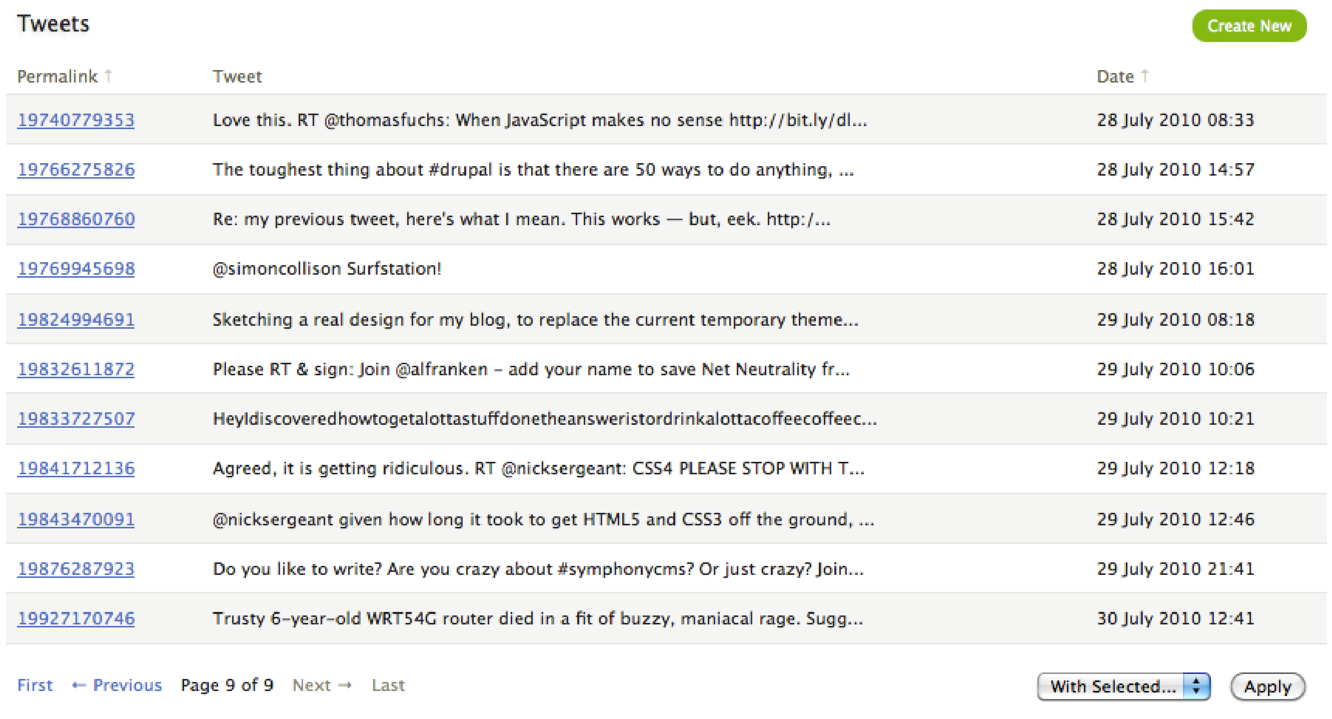
Choose: With Selected > Run > Apply. You should see something like this:


Import Complete

Import completed successfully: 147 new entries were created, 0 updated, and 0 skipped.

Woo! It worked.

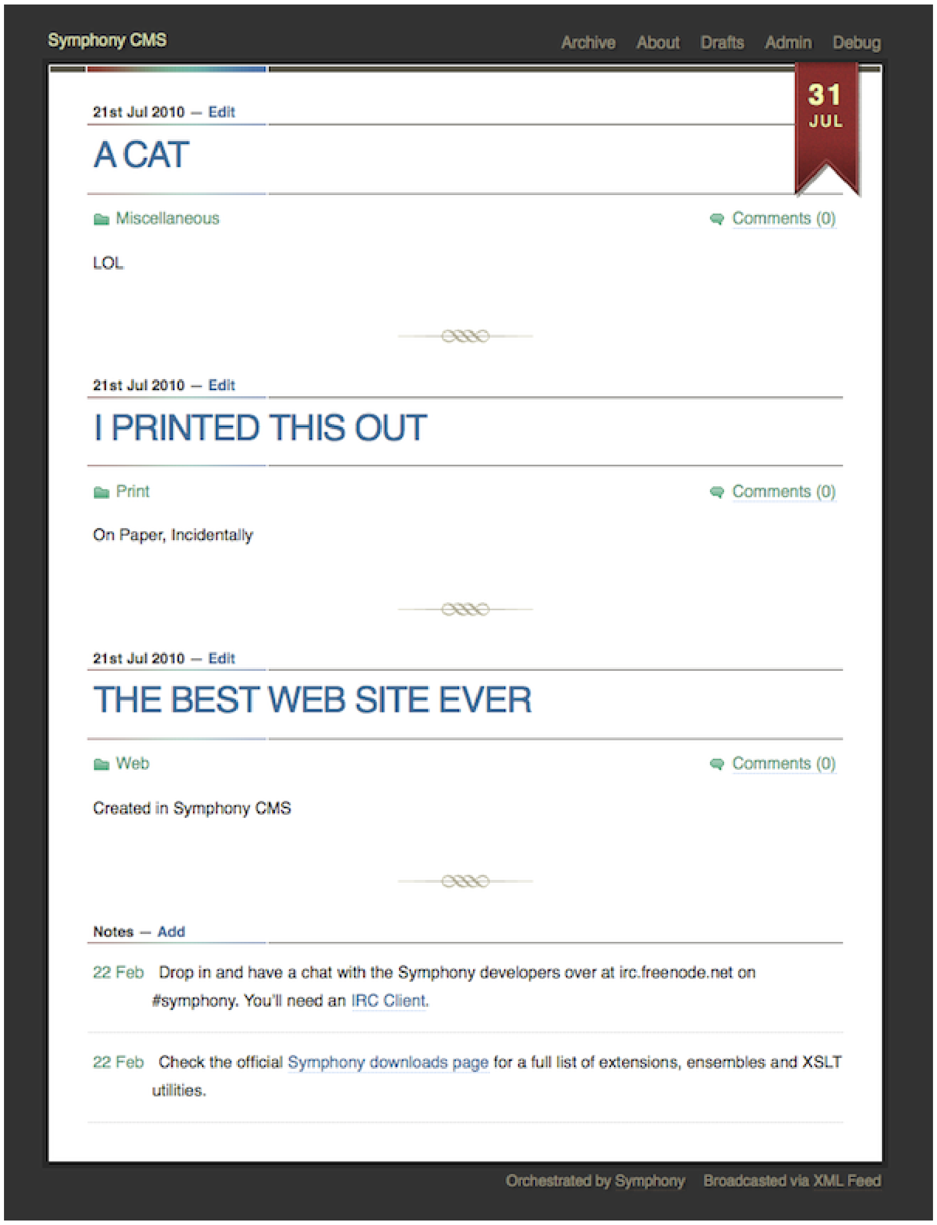
Now if you proceed to the Data > Tweets section, you will see a swath of Tweets.


Tweets

Permalink ↑	Tweet	Date ↑
19740779353	Love this. RT @thomasfuchs: When JavaScript makes no sense http://bit.ly/dl...	28 July 2010 08:33
19766275826	The toughest thing about #drupal is that there are 50 ways to do anything, ...	28 July 2010 14:57
19768860760	Re: my previous tweet, here's what I mean. This works — but, eek. http:/...	28 July 2010 15:42
19769945698	@simoncollison Surfstation!	28 July 2010 16:01
19824994691	Sketching a real design for my blog, to replace the current temporary theme...	29 July 2010 08:18
19832611872	Please RT & sign: Join @alfranken – add your name to save Net Neutrality fr...	29 July 2010 10:06
19833727507	HeyIdiscoveredhowtogetalottastuffdonetheansweristordrinkalottacoffeecoffeec...	29 July 2010 10:21
19841712136	Agreed, it is getting ridiculous. RT @nicksergeant: CSS4 PLEASE STOP WITH T...	29 July 2010 12:18
19843470091	@nicksergeant given how long it took to get HTML5 and CSS3 off the ground, ...	29 July 2010 12:46
19876287923	Do you like to write? Are you crazy about #symphonycms? Or just crazy? Join...	29 July 2010 21:41
19927170746	Trusty 6-year-old WRT54G router died in a fit of buzzy, maniacal rage. Sugg...	30 July 2010 12:41

Indeed, a swath.

Excellent! After all this work, we have content loaded in our custom sections, and some imported Tweets. Let's take a look at the site now.


Symphony CMS

Archive About Drafts Admin Debug

31 JUL

21st Jul 2010 — Edit

A CAT

Miscellaneous Comments (0)

LOL

21st Jul 2010 — Edit

I PRINTED THIS OUT

Print Comments (0)

On Paper, Incidentally

21st Jul 2010 — Edit

THE BEST WEB SITE EVER

Web Comments (0)

Created in Symphony CMS

Notes — Add

22 Feb Drop in and have a chat with the Symphony developers over at irc.freenode.net on #symphony. You'll need an IRC Client.

22 Feb Check the official Symphony downloads page for a full list of extensions, ensembles and XSLT utilities.

Orchestrated by Symphony Broadcasted via XML Feed

Demo-tastic

Still looks like a blog. Don't worry! In the next episode, we'll configure an HTML template for the site and instruct Symphony to display our content exactly as we want, using Data Sources and some XSLT mojo. We'll also get our About page and contact form in order.

Anything else you'd like to see? Leave a comment and ye shall receive.

15 COMMENTS

David Hund wrote:
Aug 1, 2010 at 10:50 am
(Edit)

Hi Jonas,

Thanks for another fine introduction to (parts of) Symphony. I'm learning Symphony myself at the moment and tutorials like this one are very valuable.

What I've noticed: I found the way in which you connect images to articles/projects confusing. Ideally you would like to add the appropriate images in a (newly created) article, not link them later on (in a separate section). This can be done, I believe, through a SBL (when you've added the images before) in article. Alternatively you could work with an extension such as Mediathek, correct?

What is the reason you link from images to articles and not the other way around. Why edit/link them apart?

As for "stuff I'd like to see discussed": 'static pages'!

One of the most confusing things of Symphony for me (and I believe many others) is the fact that, out of the box, it is not simple for editors(!) to manage (add, nest and reorder) content-pages as entries(!).

I understand (and appreciate) the fact that, in Symphony, the assumption is that the IA and site/navigation structure is worked out in advance.

However, most simple CMS'es (like Radiant, Wolf and even WordPress) allow non-technical editors to add, nest and reorder content-pages.

In Symphony this would involve creating Pages and editing XSL templates.

I've read about many ways to do this (pages as entries) but they seem to involve 'hacks' (and extension). It seems to me to be one of the main issues in 'building a website with Symphony'. Please describe your view and the way you go about doing this.

Thanks again for the time you're putting into this.

Symphony seems a wonderful CMS (-framework) and I am eager to learn more of it.

Josh N. wrote:
Aug 1, 2010 at 2:15 pm
(Edit)

Is there a way to automate the XML importer so it automagialy stays up-to-date?

I currently use a cron job to run a PHP script that does something similar, but I'd like to be able to do the same thing from the Symphony admin area.

Josh N. wrote:
Aug 1, 2010 at 2:37 pm
(Edit)

@David

I think there are 2 types of CMSs: Dynamic and Page-based. Symphony is not a page-based CMS. Even WordPress isn't page-based (even though it has some basic page functionality).

What you are asking for will have to be a special extension that does specifically what you need. Not everyone needs pages that can be nested and duplicated. What if one of those pages needs to have different fields? Different datasources? Also, not everyone wants people going in and arbitrarily adding pages to a site after spending a lot of time perfecting the IA.

You can go about making pages in Symphony a number of different ways depending on what you need. You can easily set up all of your sections and have each entry in that section be a new page. If you need to nest them, then use the select box link field and use it to choose the parent page. The hard part will be writing the XSLT and pulling the right data to build your dynamic navigation.

The point is, you and I both have different ways we want to incorporate "pages." The Symphony core CMS will not ever assume one way or another. It has been built to be lean and only contain the parts every site needs. Extra, specific, functionality is added with extensions.

In the end, there isn't one CMS that will do everything. You have to pick the right tool for the job. If your client has creating and organizing the site's pages as a high priority, Symphony is not the right CMS for that job.

Jonas wrote:
Aug 1, 2010 at 6:27 pm
(Edit)

@David: thanks for your comments!

Regarding the treatment for images, there are surely other ways to do it depending on your needs (Mediathek being one of them.)

In this case, I went with the linked sections for two reasons: 1) it's easy, as it reuses the demo workspace, and 2) treating images as entries allows you to arrange them programmatically in the site template. This will become much more clear in part 3. For the record, I would probably do things a little differently if we were building a blog (with embedded images in articles) instead of a portfolio site. If you've ever used Drupal, there is a similar issue as to whether you treat images as Nodes or as embedded content.

About static pages — that really is a whole post in itself! I've never attempted to build a Symphony site that required a significant number of pages or a complex hierarchy of pages, and for now, I probably wouldn't attempt it. As Josh said, it currently isn't the best tool for such a task, though I believe there are several developers working on a set of extensions that would provide this functionality — some of which is already available.

Jonas wrote:
Aug 1, 2010 at 6:34 pm
(Edit)

@Josh – the method for cron-tasked XML Importer I'm currently using is as follows:

1) Set up a new Symphony author that will act as a robot account. When you do this, check the "Allow remote login via…" box and make note of the random numerical string at the end of the URL.

2) In your host's crontab, use a command like so:

```
curl
http://yoursite.com/symphony/extension/xmlimporter/importers/run/tweets/?
auth-token=1234a5b6 >/dev/null 2>&1
```

where the auth-token is the piece from the URL in step 1, and "tweets" is the name of your importer.

You can do this a little nicer by using the Shell and Cron extensions as outlined in this post, but the method above has been working fine for me.

David Hund wrote:
Aug 2, 2010 at 5:28 am
(Edit)

@Jonas, @Josh thanks for the reply. I understand Symphony is not page-based and I've already mentioned it's often preferable to define a proper IA/site structure in advance.

However, I believe it is seen as a must-have for many editors/clients, even if they happen never to create new pages.

http://symphony-cms.com/discuss/thread/47932/

Anyway, even though it is not Symphony's main focus, I believe it would be very beneficial to its adoption to add this ('trivial') functionality.

I'll keep an eye out for extensions such as Page Prototypes etc.

Re: the linking of articles with images. It's clear it's often preferable to have images-as-entries. My main question here was, however, why define an 'articles' Link in images and not the other way around?

Personally, I would not have chosen to base this tutorial on the installed Ensemble but on a complete blank Symphony install. Maybe you would, in that case, choose an even simpler site, but this would have given you the freedom to build what you want, the way you want it, and it would probably have given us a bit more insight into Symphony.

Anyway, thanks again: looking forward to the rest of your tutorials!

Jonas wrote:
Aug 2, 2010 at 5:47 am
(Edit)

Agreed about pages. Wolf, Concrete5, and ExpressionEngine (with Structure extension) handle this pretty gracefully, and I'd love to see a similar implementation in Symphony.

Re: images: the reason to link images to articles, and not vice-versa, is that we want to allow multiple images to be associated with a single article, not multiple articles associated with a single image.

As to whether to use ensembles or not…I considered starting with a blank installation, but the default workspace provides a fair bit of functionality that is beneficial for a beginner, particularly with regards to XSLT templates and utilities that are preloaded.

My goal with this series was to produce a functional website with the least work possible, and using the default workspace helps get you there a little faster. But you could certainly do it leaner and meaner without the workspace, once you have some experience using the system.

Josh N. wrote:
Aug 3, 2010 at 8:03 pm
(Edit)

I would like to see several extensions that adds support for building/nesting pages as well. We have a few, but this is certainly an area that could use some more options.

The linking images to entries is a little bit hard to grasp, but the fact that you can link content like this is what makes Symphony so powerful. Not many CMSs do that.

At the Symphony Meetup, there was a presentation that showed how Airlock creates a front-end editing UI to make it easier for clients to add complex content to pages. The admin area can quickly get out of hand when you start linking a lot of content. It's worth a read and might give you some ideas: http://symphony-cms.com/discuss/thread/39550/10/#position-197

Jamie wrote:
Aug 17, 2010 at 7:22 am
(Edit)

Thanks for these brilliant tutorials to an exciting CMS, I've found them very useful. Can I expect the third part soon?

Jonas wrote:
Aug 17, 2010 at 8:58 am
(Edit)

Hi Jamie, part 3 will be up within the next week or so. Thanks for reading!

Mark wrote:
Aug 23, 2010 at 11:49 am
(Edit)

Eagerly awaiting part 3!

I'm in the process of redesigning my portfolio in Symphony, so far going well. Hoping to pick up some good practice and tips from this series of tutorials!

Jonas wrote:
Aug 23, 2010 at 12:12 pm
(Edit)

Sorry all, part 3 has gotten delayed a bit — coming soon!

Steve wrote:
Nov 6, 2010 at 12:32 am
(Edit)

Thanks for this tutorial series, this was my first intro to Symphony and it all went quite nicely… until…

When I go to XML Importers > Run, I get the following error:
Import Failed
No entries to import.

I have followed the tutorial to the letter, except that I used the provided user_id rather than specifying my own (I don't have a twitter account). I can't image this would make any difference though as I can access the feed via http://twitter.com/statuses/user_timeline.xml?user_id=16454301&count=50.

The only possible idea I have is that there appears to be a bug, when I go to edit the Tweets XML Importer, the "Automatically discover namespaces" option is unchecked, it doesn't appear to be remembering this preference.

Thoughts?

P.S. Looking forward to part 3!

Steve wrote:
Nov 6, 2010 at 12:57 am
(Edit)

just a quick follow-up – I tried out the XML Importer on a WordPress RSS feed I had handy, I saw how the namespaces are supposed to work, and the import was then successful.

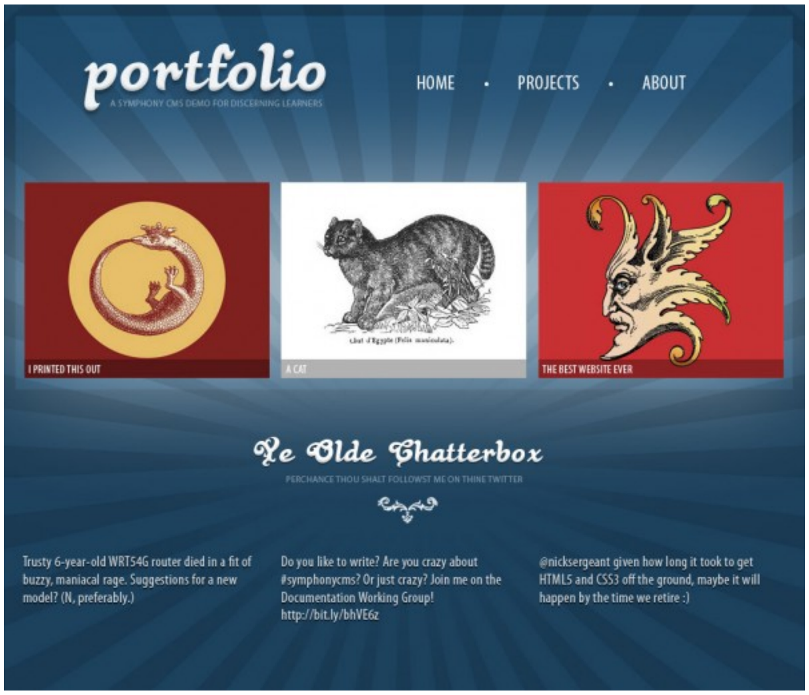
I think I can narrow the issue down just to detecting the namespaces on the twitter feed, it doesn't work for me.

Building a site with Symphony CMS, Part 3 | RocketFoo wrote:
Apr 6, 2011 at 7:42 pm
(Edit)

[…] budding Symphonists, and welcome to the long-delayed part 3 of our tutorial series! In the last episode, we did some relatively mundane work prepping the structure of our portfolio website. Part […]

BUILDING A SITE WITH SYMPHONY CMS, PART 3

Hello budding Symphonists, and welcome to the long-delayed part 3 of our tutorial series! In the last episode, we did some relatively mundane work prepping the structure of our portfolio website. Part 3's where we crank up the fun and dive into Pages, Data Sources, XML, XSLT, and maybe some other wicked sweet acronyms. Our goal is to end up with a site that looks like this:


portfolio
A SYMPHONY CMS DEMO FOR DISCERNING LEARNERS
HOME
PROJECTS
ABOUT
I PRINTED THIS OUT
A CAT
THE BEST WEBSITE EVER
Ye Olde Chatterbox
PERCHANCE THOU SHALT FOLLOWST ME ON THINE TWITTER

Trusty 6-year-old WRT54G router died in a fit of buzzy, maniacal rage. Suggestions for a new model? (N, preferably.)

Do you like to write? Are you crazy about #symphonycms? Or just crazy? Join me on the Documentation Working Group! http://bit.ly/bhVE6z

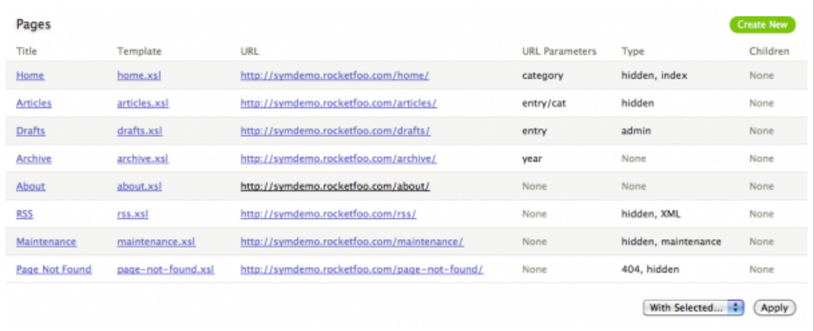
@nicksergeant given how long it took to get HTML5 and CSS3 off the ground, maybe it will happen by the time we retire :)

Before I move on to the action, just wanted to mention that several versions of Symphony have been released since I (long ago) began this series. As of this writing, the current version is 2.2. It's fantastic, go get it.

GETTING STARTED WITH PAGES AND TEMPLATES

OK, let's begin by getting our rough template in place. If you'll recall, back in part 1, we preloaded an XSLT Utility called *html5.xsl* (in case you forgot, it's here.) This is a great master template that gives us a standard HTML5 layout marked up in XSLT and ready for use in Symphony.

So how do you use this thing? Take a look at our Symphony pages by going to Blueprints > Pages.



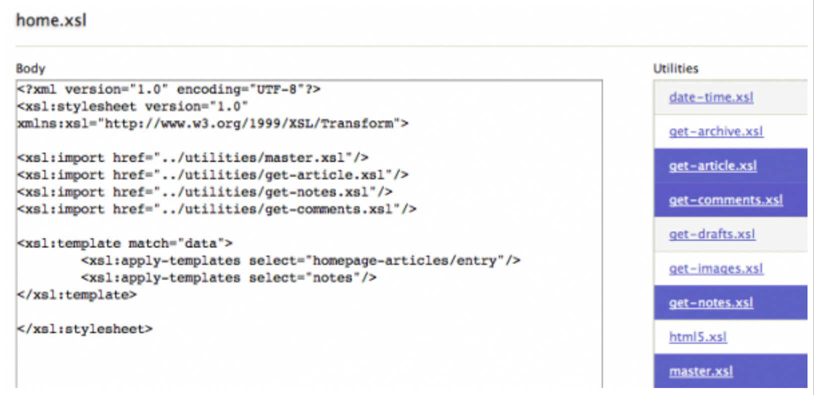
Title	Template	URL	URL Parameters	Type	Children
Home	home.xsl	http://symdemo.rocketfoo.com/home/	category	hidden, index	None
Articles	articles.xsl	http://symdemo.rocketfoo.com/articles/	entry/cat	hidden	None
Drafts	drafts.xsl	http://symdemo.rocketfoo.com/drafts/	entry	admin	None
Archive	archive.xsl	http://symdemo.rocketfoo.com/archive/	year	None	None
About	about.xsl	http://symdemo.rocketfoo.com/about/	None	None	None
RSS	rss.xsl	http://symdemo.rocketfoo.com/rss/	None	hidden, XML	None
Maintenance	maintenance.xsl	http://symdemo.rocketfoo.com/maintenance/	None	hidden, maintenance	None
Page Not Found	page-not-found.xsl	http://symdemo.rocketfoo.com/page-not-found/	None	404, hidden	None

P-P-P-Pages

As you can see, we already have several pages, .xsl templates, and corresponding URLs in our new site. These came with the default workspace. To be clear, pages in Symphony are not really “pages” in the traditional sense. Says the documentation, “pages are not simply containers for static content and there is not a one-to-one relationship between a page and the content accessible via that page. A single Symphony page can power an entire browsing interface.”

Not surprisingly, pages are the bread and butter of your site, tying together your content, templates, and user interface elements. (Don’t worry, this will make more sense in a minute.)

Anyway, each page has its own .xsl template file. This allows you to customize how that page’s content is displayed in a browser. Take a look at our home page template by selecting the *home.xsl* link.

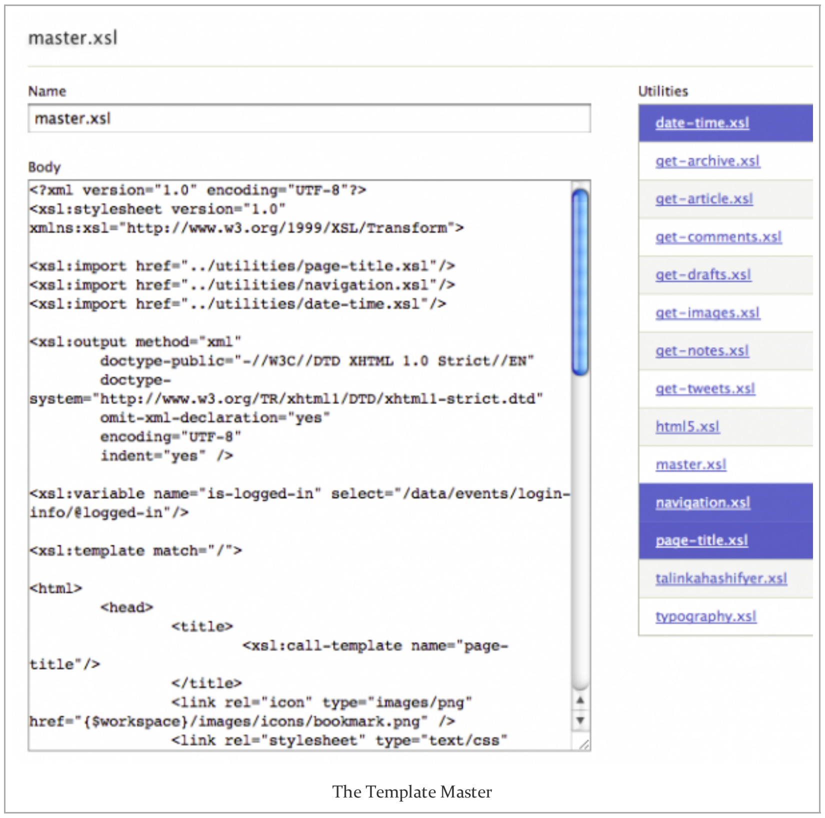


```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:import href="../utilities/master.xsl"/>
<xsl:import href="../utilities/get-article.xsl"/>
<xsl:import href="../utilities/get-notes.xsl"/>
<xsl:import href="../utilities/get-comments.xsl"/>

<xsl:template match="data">
        <xsl:apply-templates select="homepage-articles/entry"/>
        <xsl:apply-templates select="notes"/>
</xsl:template>

</xsl:stylesheet>
```

- date-time.xsl
- get-archive.xsl
- get-article.xsl
- get-comments.xsl
- get-drafts.xsl
- get-images.xsl
- get-notes.xsl
- html5.xsl
- master.xsl

Home is where the XSLT is

Not much to it, eh? Almost all of this page’s layout and content is arranged and formatted by other XSLT utilities that are imported to this template; the relevant imported utilities are highlighted in blue in the right column. In that list, click on master.xsl.



master.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:import href="../utilities/page-title.xsl"/>
<xsl:import href="../utilities/navigation.xsl"/>
<xsl:import href="../utilities/date-time.xsl"/>

<xsl:output method="xml"
        doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
        doctype-
system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
        omit-xml-declaration="yes"
        encoding="UTF-8"
        indent="yes" />

<xsl:variable name="is-logged-in" select="/data/events/login-
info/@logged-in"/>

<xsl:template match="/">

<html>
        <head>
                <title>
                        <xsl:call-template name="page-
title"/>
                </title>
                <link rel="icon" type="images/png"
href="{$workspace}/images/icons/bookmark.png" />
                <link rel="stylesheet" type="text/css"
```

The Template Master

Aha! This is the primary site design template, which can be imported into all of our main pages. This template provides the standard elements of a site design: header, nav, etc. (But to our horror, this master.xsl is old-school XHTML, which was excellent about one year ago but is suddenly the laughingstock of fashionable web designers.)

Let's hang on to this master.xsl for now, and switch our home page to our preferred template, html5.xsl. Go back to the home.xsl template and change this line:

~~`<xsl:import href="../utilities/master.xsl"/>`~~

```
<xsl:import href="../utilities/html5.xsl"/>
```

Let's also get rid of some bits we don't need. We're not going to use the Notes or Comments functions from the default installation, so remove these lines:

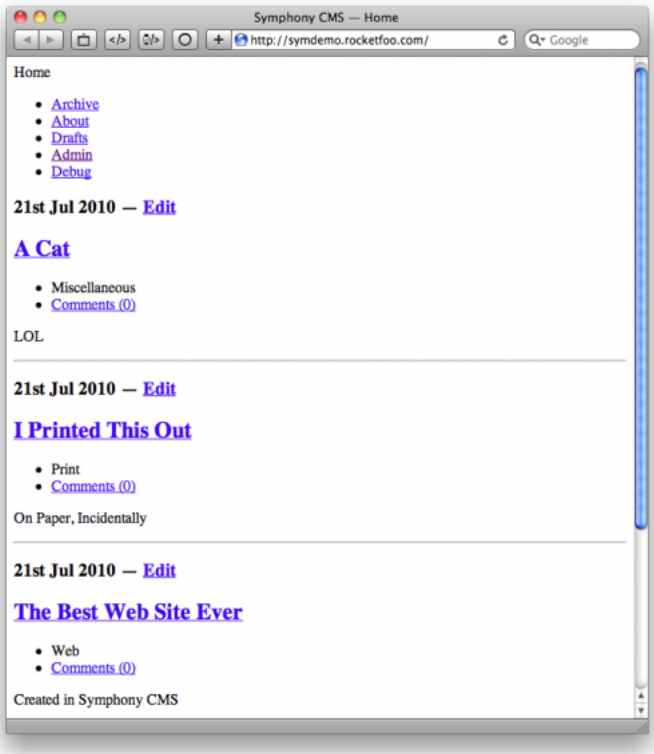
```
<xsl:import href="../utilities/get-notes.xsl"/>
<xsl:import href="../utilities/get-comments.xsl"/>
<xsl:apply-templates select="notes"/>
```

Save your changes. For our last bit of work in this area, click on the html5.xsl utility and peruse its code. Let's start over with styling by modifying our stylesheet name:

```
<link rel="stylesheet" media="screen" href="
{$workspace}/css/styles.css" />
```

change *styles.css* to "demostyles.css." This file doesn't exist yet, so we will initially have no CSS styles applied to our site.

OK, save the changes. Now to see what we've done:


Symphony CMS — Home

http://symdemo.rocketfoo.com/

Home

- Archive
- About
- Drafts
- Admin
- Debug

21st Jul 2010 — Edit

A Cat

- Miscellaneous
- Comments (0)

LOL

21st Jul 2010 — Edit

I Printed This Out

- Print
- Comments (0)

On Paper, Incidentally

21st Jul 2010 — Edit

The Best Web Site Ever

- Web
- Comments (0)

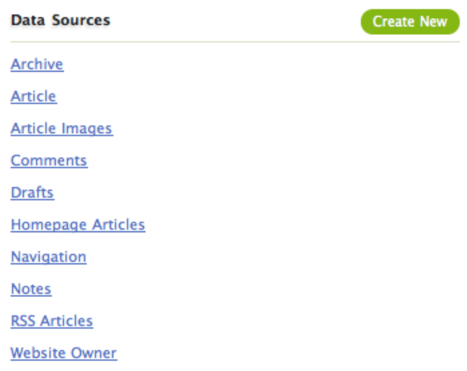
Created in Symphony CMS

1996 called and wants it website back.

Hmm, ugly. But we're not done yet. Our next matter of business is to make sure all the content we want is available on the home page.

DATA SOURCES

In Symphony, *Data Sources* are a method to select content from sections and make it available in pages. Remember, we defined several sections — Projects, Categories, Images, and Tweets — but only a fraction of this content is currently visible on our home page. Let's fix that. Go to Blueprints > Components, and have a look at the Data Sources list.

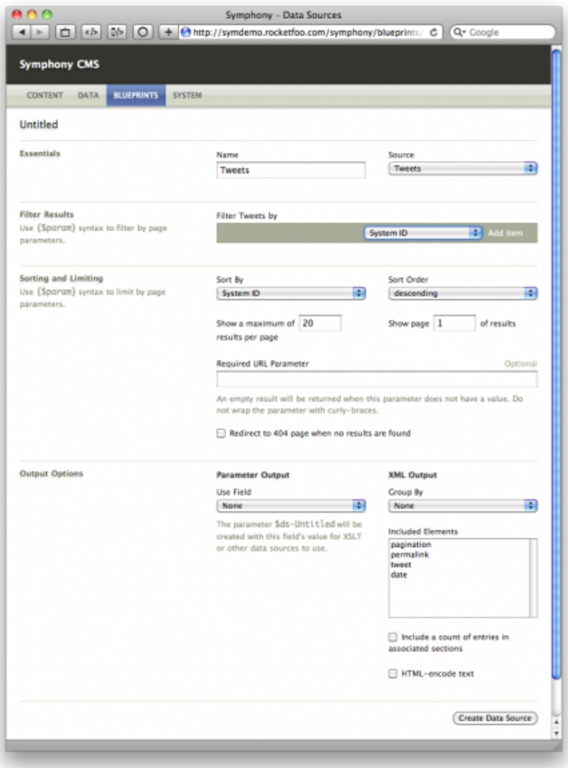

Data Sources

Create New

- Archive
- Article
- Article Images
- Comments
- Drafts
- Homepage Articles
- Navigation
- Notes
- RSS Articles
- Website Owner

A Fountain of Data

As usual, there are some pre-made Data Sources here, and we can reuse most of them. Since there is no Data Source for our Tweets, let's make one. Select *Create New*.



Untitled

Essentials

Name: Tweets
Source: Tweets

Filter Results
Use {$param} syntax to filter by page parameters.

Filter Tweets by: System ID — Add item

Sorting and Limiting
Use {$param} syntax to limit by page parameters.

Sort By: System ID
Sort Order: descending
Show a maximum of 20 results per page
Show page 1 of results

Required URL Parameter (Optional)

An empty result will be returned when this parameter does not have a value. Do not wrap the parameter with curly-braces.

☐ Redirect to 404 page when no results are found

Output Options

Parameter Output
Use Field: None
The parameter $ds-Untitled will be created with this field's value for XSLT or other data sources to use.

XML Output
Group By: None
Included Elements: pagination, permalink, tweet, date

☐ Include a count of entries in associated sections

☐ HTML-encode text

Create Data Source

Creating a Data Source For Some Homestyle Tweetin's

this can look a little intimidating at first, but it's actually not so bad. Start out by naming this Data Source "Tweets" and select the *Tweets* section from the Source menu.

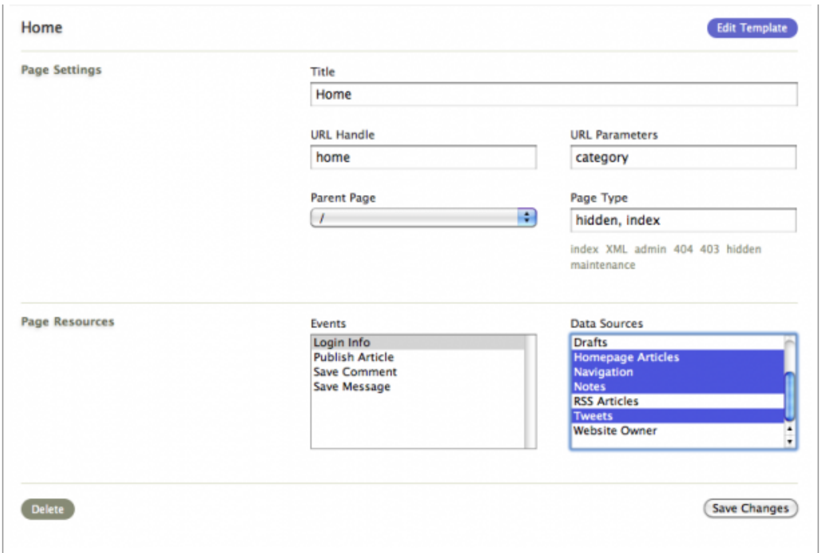
The next step to consider is whether you want to **Filter** your Tweets by some criteria. For now we'll leave this alone, but one example I use on my site is to hide any @-replies to specific people, using this filter: `not-regexp:^@`. *Note: not-regexp is new in Symphony 2.2.*

Next, we'll have to decide how we want to **Sort and Limit** our Tweets. I would like them to be in descending date order (newest at the top) and I don't want to see more than three Tweets on the home page. So, change **Sort By** to *Date* and change "Show a maximum of 20 results per page" to 3.

In the **Included Elements** select list, choose *permalink*, *tweet*, and *date*. These are the pieces of a Tweet that will be available to your page.

We're done for now — select *Create Data Source*.

Wait, we're not done! We need to instruct our home page that this new Data Source is available. Go back to Blueprints > Pages, and click on the *Home* link in the **Title** column. This brings up the Page Settings screen for the home page.


Home

Edit Template

Page Settings

Title: Home

URL Handle: home

URL Parameters: category

Parent Page: /

Page Type: hidden, index

index XML admin 404 403 hidden maintenance

Page Resources

Events: Login Info, Publish Article, Save Comment, Save Message

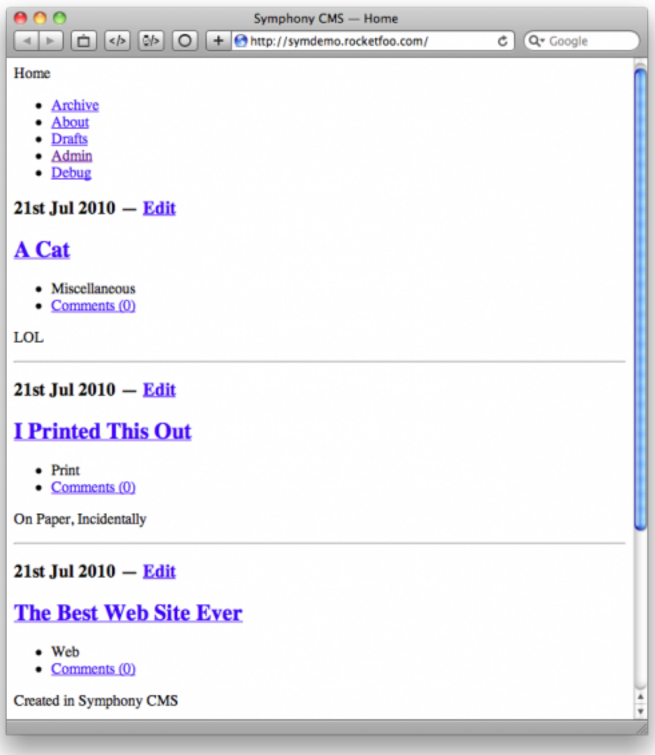
Data Sources: Drafts, Homepage Articles, Navigation, Notes, RSS Articles, Tweets, Website Owner

Delete

Save Changes

Setting Page Settings

Command or Control -click on the **Tweets** Data Source to make sure it's selected, and then *Save Changes*. Now let's have another look at our home page.


Symphony CMS — Home

http://symdemo.rocketfoo.com/

Home

- Archive
- About
- Drafts
- Admin
- Debug

21st Jul 2010 — Edit

A Cat

- Miscellaneous
- Comments (0)

LOL

21st Jul 2010 — Edit

I Printed This Out

- Print
- Comments (0)

On Paper, Incidentally

21st Jul 2010 — Edit

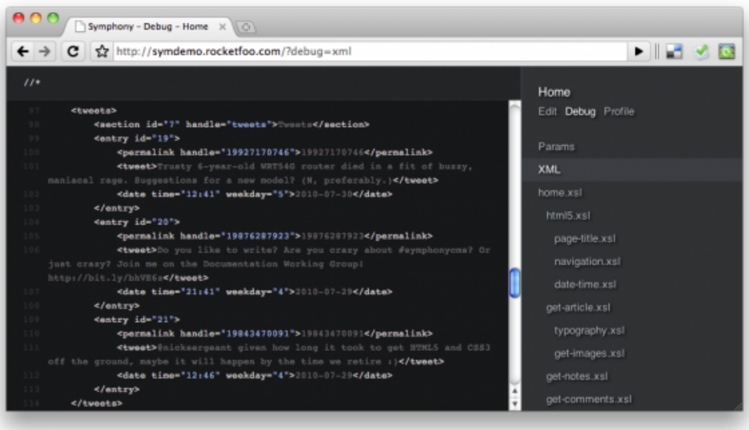
The Best Web Site Ever

- Web
- Comments (0)

Created in Symphony CMS

Tweet?

As you can see, our Tweets are all here. What's that? They're not? Here's why: the home page now has access to our Tweets data, but we haven't added any Tweet formatting to our front-end templates yet. Don't believe me? Click the *Debug* link or append `?debug=xml` to the end of the URL, and begin to understand the glory that is Symphony.



```
    <tweets>
        <section id="7" handle="tweets">Tweets</section>
        <entry id="19">
            <permalink handle="19927170746">19927170746</permalink>
            <tweet>Trusty 6-year-old WRT54G router died in a fit of buzzy,
maniacal rage. Suggestions for a new model? (N, preferably.)</tweet>
            <date time="12:41" weekday="5">2010-07-30</date>
        </entry>
        <entry id="20">
            <permalink handle="19876287923">19876287923</permalink>
            <tweet>Do you like to write? Are you crazy about #symphonycms? Or
just crazy? Join me on the Documentation Working Group!
http://bit.ly/bhVE6s</tweet>
            <date time="21:41" weekday="4">2010-07-29</date>
        </entry>
        <entry id="21">
            <permalink handle="19843470091">19843470091</permalink>
            <tweet>@nicksergeant given how long it took to get HTML5 and CSS3
off the ground, maybe it will happen by the time we retire :)</tweet>
            <date time="12:46" weekday="4">2010-07-29</date>
        </entry>
    </tweets>
```

Oh, the sweet sweet Tweet glory.

Now you're seeing the output of all the Data Sources enabled for this page, rendered as XML. We're free to use XSLT to transform this into whatever format we want (in this case it's HTML, but it doesn't have to be.) So let's do that. There are many ways to approach this, so I'll give you one example, but you can try other variations too.

A SPOT OF XSLT

Go to Blueprints > Components and in the Utilities section, select *Create New*. We're going to name this utility *get-tweets.xsl*. Paste in the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:import href="../utilities/talinkahashifyer.xsl"/>

    <xsl:template name="get-tweets" match="tweets">
      <ul>
        <xsl:apply-templates select="entry"/>
      </ul>
      </xsl:template>

    <xsl:template match="tweets/entry">
      <li>
        <xsl:call-template name="linkahashify">
          <xsl:with-param name="tweet" select="tweet" />
        </xsl:call-template>
      </li>
    </xsl:template>
</xsl:stylesheet>
```

(I'll step through this briefly, but be sure to check out the metric ton of information about XSLT out there.)

Our Tweets XML looks like this:

```
    <tweets>
        <section id="7" handle="tweets">Tweets</section>
        <entry id="19">
            <permalink handle="19927170746">19927170746</permalink>
            <tweet>Trusty 6-year-old WRT54G router died in a fit of buzzy, maniacal rage. Suggestions for a new model? (N, preferably.)</twe
            <date time="12:41" weekday="5">2010-07-30</date>
        </entry>
        <entry id="20">
            <permalink handle="19876287923">19876287923</permalink>
            <tweet>Do you like to write? Are you crazy about #symphonycms? Or just crazy? Join me on the Documentation Working Group! http:/
            <date time="21:41" weekday="4">2010-07-29</date>
        </entry>
        <entry id="21">
            <permalink handle="19843470091">19843470091</permalink>
            <tweet>@nicksergeant given how long it took to get HTML5 and CSS3 off the ground, maybe it will happen by the time we retire :)<
            <date time="12:46" weekday="4">2010-07-29</date>
        </entry>
    </tweets>
```

So here's how the XSLT works:

`<xsl:import href="../utilities/talinkahashifyer.xsl"/>`
Imports the "talinkahashifyer" utility we downloaded from the Symphony site — this auto-formats the body of our tweets so things like links and @replies work.

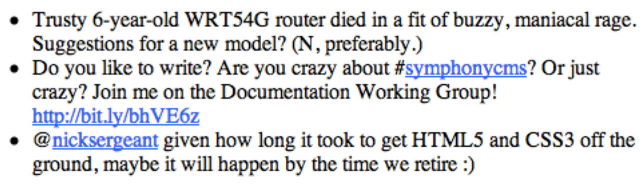
```
<xsl:template name="get-tweets" match="tweets">
    <ul>
        <xsl:apply-templates select="entry"/>
    </ul>
</xsl:template>
```

This selects the `<tweets>` parent element from our XML, and applies any relevant templates for the `<entry>` child elements. The output is contained in an unordered list (ul) tag.

```
<xsl:template match="tweets/entry">
    <li>
        <xsl:call-template name="linkahashify">
            <xsl:with-param name="tweet" select="tweet" />
        </xsl:call-template>
    </li>
</xsl:template>
```

This matches all `<entry>` elements contained in the `<tweets>` element, and formats them by calling the linkahashify template (the actual tweet content is sent as a parameter to that template.) This output will be contained in a list (li) element.

For our example, the end result will be an unordered list of three tweets, organized in descending date order, and formatted appropriately:



- Trusty 6-year-old WRT54G router died in a fit of buzzy, maniacal rage. Suggestions for a new model? (N, preferably.)
- Do you like to write? Are you crazy about #symphonycms? Or just crazy? Join me on the Documentation Working Group! http://bit.ly/bhVE6z
- @nicksergeant given how long it took to get HTML5 and CSS3 off the ground, maybe it will happen by the time we retire :)

This solution is admittedly somewhat abstract, but it's good to familiarize yourself with `<xsl:apply-templates>` and `<xsl:call-template>`, as they are used heavily. Here's a more straightforward way to do this with `<xsl:for-each>`, which might be more obvious for XSLT newbies who are familiar with other template languages:

```
<xsl:template match="tweets">
 <ul>
    <xsl:for-each select="entry">
      <li>
        <xsl:call-template name="linkahashify">
          <xsl:with-param name="tweet" select="tweet" />
        </xsl:call-template>
      </li>
    </xsl:for-each>
</ul>
```

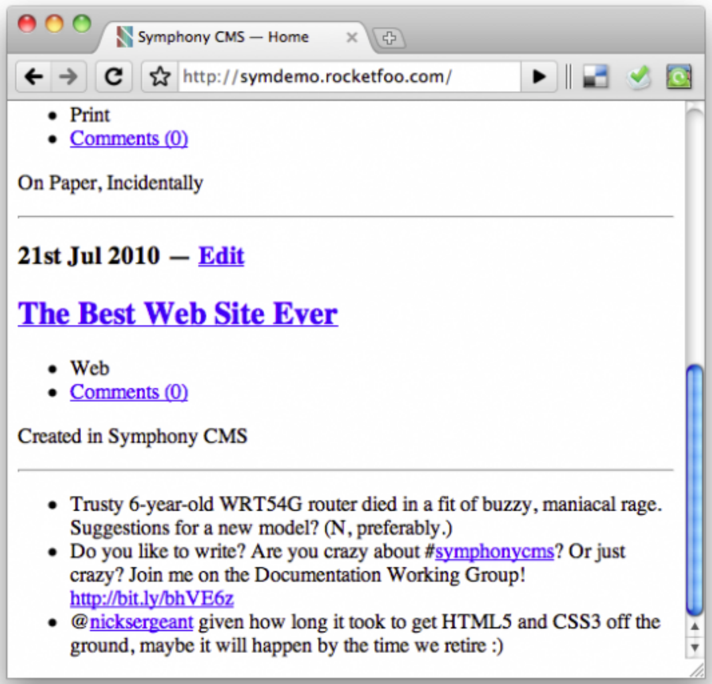
```
</xsl:template>
```

Now that we have the *get-tweets.xsl* utility set up, let's adjust our home page template to use it. Go to Blueprints > Pages and click on the home.xsl link.

We need to do two things: import *get-tweets.xsl*, and apply the tweets template to the content for this page. The complete home.xsl is below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:import href="../utilities/html5.xsl"/>
<xsl:import href="../utilities/get-article.xsl"/>
<xsl:import href="../utilities/get-tweets.xsl"/>

<xsl:template match="data">
  <xsl:apply-templates select="homepage-articles/entry"/>
  <xsl:apply-templates select="tweets"/>
</xsl:template>

</xsl:stylesheet>
```

Woo, now let's check the page one more time.



- Print
- Comments (0)

On Paper, Incidentally

21st Jul 2010 — Edit

The Best Web Site Ever

- Web
- Comments (0)

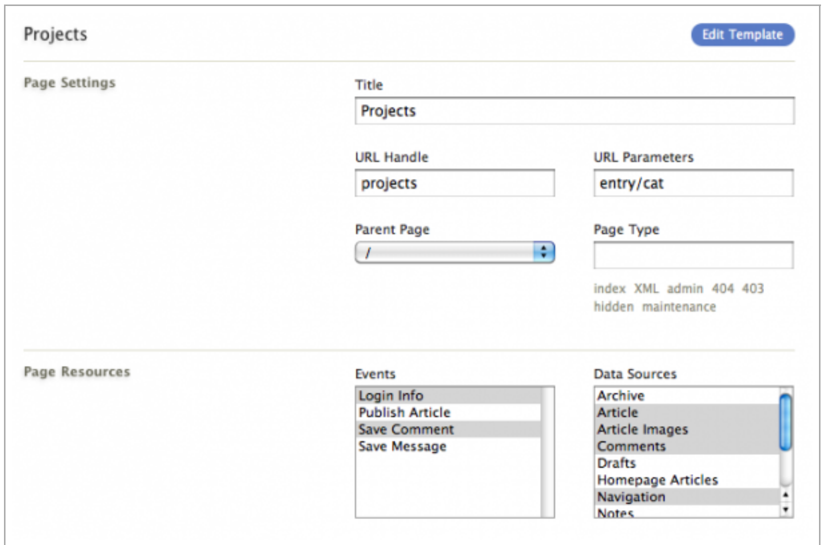
Created in Symphony CMS

- Trusty 6-year-old WRT54G router died in a fit of buzzy, maniacal rage. Suggestions for a new model? (N, preferably.)
- Do you like to write? Are you crazy about #symphonycms? Or just crazy? Join me on the Documentation Working Group! http://bit.ly/bhVE6z
- @nicksergeant given how long it took to get HTML5 and CSS3 off the ground, maybe it will happen by the time we retire :)

Tweets are go

For our last content trick, we need to display some thumbnail images for each project. Before we do that, let's clean up and tweak our setup a bit.

First, go to the Articles page and rename it *Projects*. We want this page to be available to the navigation, so remove the *hidden* page type.


Projects

Edit Template

Page Settings

Title: Projects

URL Handle: projects

URL Parameters: entry/cat

Parent Page: /

Page Type:

index XML admin 404 403 hidden maintenance

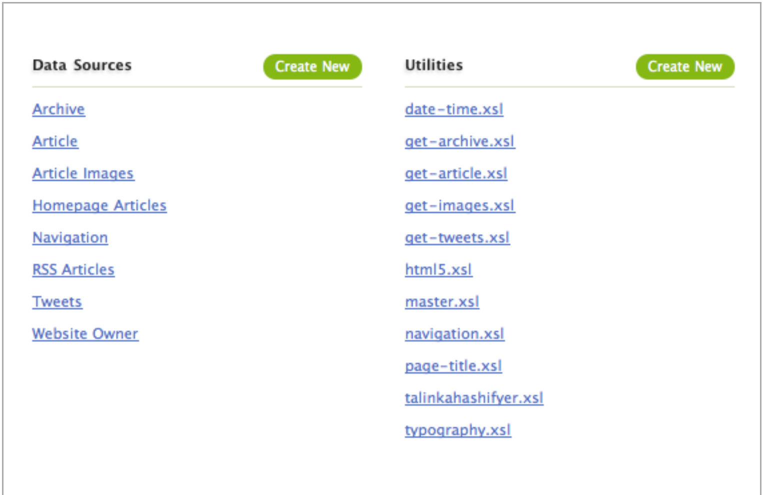
Page Resources

Events: Login Info, Publish Article, Save Comment, Save Message

Data Sources: Archive, Article, Article Images, Comments, Drafts, Homepage Articles, Navigation, Notes

Projectifying

Next, delete Comments, Notes, and Drafts stuff from the Blueprints > Components and Blueprints > Sections areas.


Data Sources | Create New

Archive
Article
Article Images
Homepage Articles
Navigation
RSS Articles
Tweets
Website Owner

Utilities | Create New

date-time.xsl
get-archive.xsl
get-article.xsl
get-images.xsl
get-tweets.xsl
html5.xsl
master.xsl
navigation.xsl
page-title.xsl
talinkahashifyer.xsl
typography.xsl

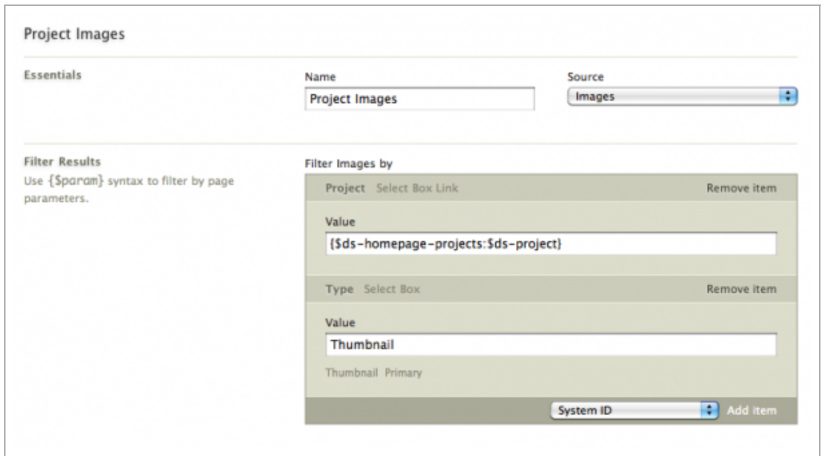
Clean as a whistle

For my last bit of cleanup, I'm going to replace all instances of the word 'article' with the word 'project' in the remaining XSLT page templates, utilities, and data sources. This is mostly for looks, you could skip this. Note that there are several ways to go about this sort of thing, you can either edit the files via the Symphony interface (slower) or operate on them as files in `$site-root/workspace` (faster).

DEALING WITH IMAGES

Now that things are tidy, let's work on getting images on the home page. There are three items that need to be tweaked: the Project Images data source, and some XSLT utilities.

We only want to display thumbnails on the home page and projects page, so in the Project Images data source, add a **Filter** based on the Image Type, and set it to *Thumbnail*.


Project Images

Essentials

Name: Project Images

Source: Images

Filter Results

Use {$param} syntax to filter by page parameters.

Filter Images by

Project Select Box Link — Remove item

Value: {$ds-homepage-projects:$ds-project}

Type Select Box — Remove item

Value: Thumbnail

Thumbnail Primary

Bring out your thumbnails

Next, we'll make some minor changes to the *get-images.xsl* template. We're mainly removing the default JIT resizing code, because our thumbnails are already the size we want. Full template code here:

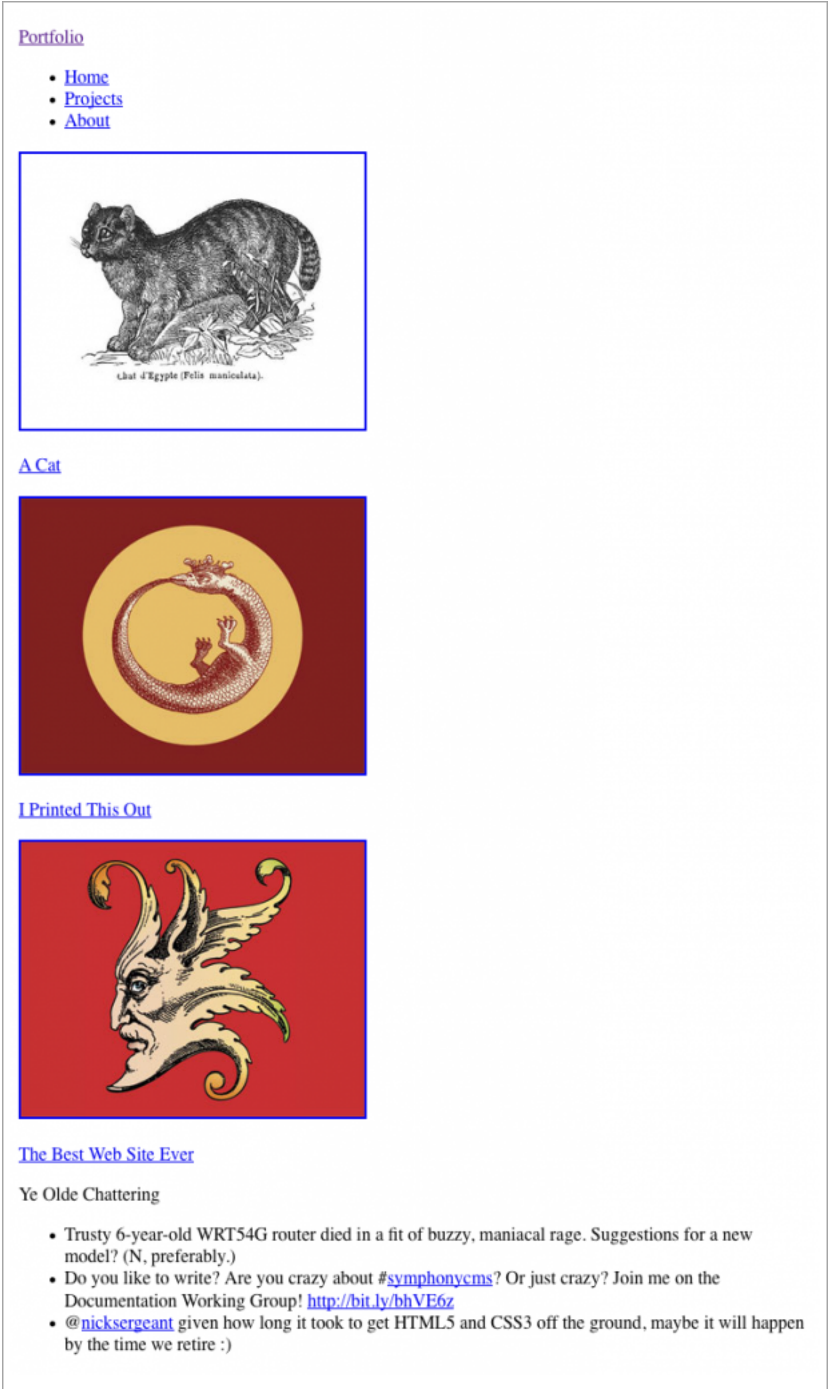
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template name="get-images">
  <xsl:param name="entry-id"/>
  <xsl:if test="/data/project-images/entry[project/item/@id = $entry-id]">
      <xsl:apply-templates select="/data/project-images/entry[project/item/@id = $entry-id]"/>
  </xsl:if>
</xsl:template>

<xsl:template match="project-images/entry">
  <a href="{$workspace}/uploads/{image/filename}">
    <xsl:if test="position() mod 3 = 0">
      <xsl:attribute name="class">last-column</xsl:attribute>
    </xsl:if>
    <img title="{description}">
      <xsl:attribute name="src">
        <xsl:value-of select="$root"/>
        <xsl:text>/workspace/uploads/</xsl:text>
        <xsl:value-of select="image/filename"/>
      </xsl:attribute>
    </img>
  </a>
</xsl:template>
</xsl:stylesheet>
```

Finally, we'll update the `homepage-projects/entry` portion of the *get-project.xsl* template. This is just a matter of simplifying what came with the default workspace.

```
<xsl:template match="homepage-projects/entry">
  <div class="entry">
    <xsl:call-template name="get-images">
      <xsl:with-param name="entry-id" select="@id"/>
    </xsl:call-template>
    <p class="imgcaption">
      <a href="{$root}/projects/{title/@handle}/"><xsl:value-of select="title"/></a>
      <xsl:if test="$is-logged-in = 'true'">
        <xsl:text> &#8212; </xsl:text>
        <a class="edit" href="{$root}/symphony/publish/{../section/@handle}/edit/{@id}/">Edit</a>
      </xsl:if>
    </p>
  </div>
</xsl:template>
```

this code simply wraps each thumbnail in a `<div>`, with its title displayed as a caption below in a `<p>` and a handy *Edit* link if you're logged into Symphony as an editor.

Phew, that was a lot of work, but our home page is finally where we want it:



Portfolio

- Home
- Projects
- About

A Cat

I Printed This Out

The Best Web Site Ever

Ye Olde Chattering

- Trusty 6-year-old WRT54G router died in a fit of buzzy, maniacal rage. Suggestions for a new model? (N, preferably.)
- Do you like to write? Are you crazy about #symphonycms? Or just crazy? Join me on the Documentation Working Group! http://bit.ly/bhVE6z
- @nicksergeant given how long it took to get HTML5 and CSS3 off the ground, maybe it will happen by the time we retire :)

Portfolio home page in the raw

We're close to the finish line now! I'm going to end this series with a short Part 4, in which I'll add the About and Projects pages, and some styles to pretty everything up. You will also be able to download the sample site as an ensemble.

P.S. I won't take so long for the Part 4, I promise

7 COMMENTS

zabba wrote:
Apr 8, 2011 at 3:52 pm
(Edit)

Jonas, this couldn't be more welcome than now as I'm trying to get my head around Symphony! I love Symphony. It gets you there after going thru the usual cmses like WP, TXP, EE etc. But not before. I love the minimalistic approach and the granular control you get with Symphony. Great post! Thank you so much!

Jonas wrote:
Apr 8, 2011 at 7:15 pm
(Edit)

Glad it helped, man! Good luck with your project.