# Getting Started with dotCMS

# 1. CREATING YOUR FIRST SITE

## *1.1. Concepts: Templates, Containers and Content*

In dotCMS, creating a new template and pages based on those templates is an extremely straightforward process. Before we begin, there are a couple of basic concepts of dotCMS that are important to understand – like what is a template, what is a container and what is content. We will also discuss how they should be brought together to create web pages and a website.

### 1.1.1. Templates

To begin with, a template in dotCMS is really just a layout for delivering an HTML, Text or XML page. If you have any experience with building websites, then templates will look familiar. They look like plain text, html or xml files that have a little bit of special markup in them to include containers or content as needed. When building a new page in dotCMS, the user must select a template in order to render the page. When building a new template, a user can click a button an browse the list of containers to select the one they would like to include.

### 1.1.2. Containers

Inside of templates, dotCMS uses containers. Containers are included in templates to format and display content items within pages. The simplest way to describe containers is as server side-include files with additional features. If you have any experience with using include files to build a web site, then you will understand how to use containers within templates. Containers are also more powerful than server side includes. Containers are in charge of formatting and displaying site content in various ways. You can create a container to show a single piece of content, or you can create a container to show lists of content. CMS Users can order these lists of content dynamically or CMS users can choose to order the content manually on the page. Containers are also important because they allow a user to see the "edit controls" when previewing a page in "preview mode".

### 1.1.3. Content

Finally, there is content. Content in dotCMS does not live on html pages or within files or templates. Content in dotCMS lives in a content repository, a big bucket of content that can be searched, added-to and reused as needed across your web site or even across different websites. Using different *containers*, the same piece of content can be displayed in different ways. For example, let us say we have a new press release content item. The home page template might have a container that only shows the title, date and teaser of press releases in a list. A user might be able to click on the press release and be taken to a detail page that has a container on it that shows the full press release details, including author, subtitle, body etc…

Let's take a closer look at the concepts of templates, containers and content, and see how we can start to create a robust, fully content managed site with the dotCMS system.

## *1.2. Getting started*

When you first set up a clean install of the dotCMS system, it will automatically create two common types of content for the content repository – Web Page Content and News Item. The system will also create a default host that we can use to begin building our site. On logging into the dotCMS system the first time, you will be presented with a tabbed interface. Click on the tab called "Website", which is where you can mange your site's folder structure, pages, files, containers and templates that make up your site.

### 1.2.1. Demo HTML

For demonstration purposes, we have created a simple html page that we will use as our starting point. While this is a simple page, dotCMS will accept any crazy html or xml page you can create. Here is the HTML of the page:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
    <head>
    <style type="text/css">
           body{
                font-family: verdana, helvetica, tahoma;
                text-align:center;
           }
           td{
                font-size: 80%;
           }
           #mainCSS{
               border:1px solid gray;
               height:600px;
               width: 800px;
           }
           #headerCSS{
                   height:70px;
                   background: #eee;
           }
           #footerCSS{
                height:70px;
                background: #eee;
            }
            #navigationCSS{
                width:120px;
                background: #eee;
                vertical-align:top;
            }
            #bodyCSS{
                background: #fff;
                vertical-align:top;
            }
        </style>

        <title>$pageTitle</title>
        </head>
        <body>
            <table id="mainCSS">
                <tr id="headerCSS">
                    <td colspan="2">
                        HEADER CONTAINER HERE
                    </td>
                </tr>
                <tr>
                    <td id="navigationCSS">
                        NAVIGATION CONTAINER HERE
                    </td>
                    <td id="bodyCSS">
                        BODY CONTAINER HERE
                    </td>
                </tr>
                <tr id="footerCSS">
                    <td colspan="2">
                        FOOTER CONTAINER HERE
                    </td>
                </tr>
            </table>
        </body>
    </html>
```

### 1.2.2. Step 1: Creating Containers

The first thing we do is create a few simple containers that will act as common headers, navigation and footers. These containers will not be responsible for holding or formatting any content from our content repository and will act as simple include files. We start by navigating to the "website" tab and then click on the sub tab called "Containers". From there, we click on "add container".

*Header Container*

The first container we are going to add is "header container". For this demo, this container will not do anything, though we could add the ability for a user to select a header or hero image. For now, let's give the container a descriptive title and then cut and paste our header HTML:

```
<h1>$pageTitle</h2>
```

into the box and hit save and publish. The `$pageTitle` variable will automatically be replaced by the page title that we enter when we create any new web page.

*Left Hand Navigation Container*
The second container we are going to add we will call the "left-hand nav" container. This container will call a macro that will automatically build the site navigation for us:

```
#navigation(1 2)
```

This macro says build an unordered list based on the page and folder structure starting from level 1 and going 2 levels deep. dotCMS has a number of **macros** (http://www.dotcms.org/documentation/tips_tricks.dot) that can help you to build websites. For example, there is a photo gallery macro or a MP3 player macro. We will talk about macros in another document. For now, the navigation macro we will use takes two parameters – the level to start building navigation from and the depth of the navigation. For this example, we want to show the top-level pages and any pages, folders or links 2 levels deep.

*Footer Container*
The third container we are going to create is the footer container. This container will really do nothing and will hold boilerplate footer information.

```
my footer &copy2007
```

*Body Container*
The fourth container we will add is the body container. This is container will represent the main content area on any page that uses it. To create this container we hit add container again and add the descriptive title "body container". We then tell the container that it will be displaying contentlets. We do this by changing the max contentlet value to 1. This means we can show up to 1 piece of content in this container.

We then select the type of content we want to show in the container. Containers can show any type of content in the content repository but for our purposes, let's select the generic "Web Page Content".

We will skip over the "Dynamic" containers for now (we will explain them in another document).

Containers are designed to show one or more pieces of content by looping over the content associated with the container. The "preloop" field is where we would put any html markup or formatting code that we would like to see run before we loop over our content. This could open a <div>, a <table> for tabular information, or even an opening xml tag. For now, let's leave it blank.

The "code" field is where we can format and display content from the content repository in this container. We do this by adding HTML markup and variables that will be replaced by fields from our content type when the page is rendered. The content type we selected above, "Web Page Content" is a very simple content type with only two fields – a title and a body. We can see the variables used to access these fields by clicking on the "add variable" button. The "add variable" button also allow you to access the content identity field. The content identity field can be passed as a variable to specific what content to display.

From this window, we can select the fields we would like to display in our container. Let's add the title and the body to our container. To format the code, let's add a <h2> tag around the title and a <p> tag around the body, so the code looks like this:

```
<h2>$!{ContentletTitle}</h2>
<p>$!{Body}</p>
```

Save the container. Now we have four containers that we can use to build a template for our website.

### 1.2.3. Step 2: Creating Templates

Click on the "templates" tab in the website browser then click on the add template button. Let's give our template a creative and descriptive name. "Body Page Template". As an aside, you can also upload a screen capture image of the template to allow CMS users to preview the template design.

Let's take our template HTML from above (the demo html) and cut and paste it directly into template box. Now, all we have to do is replace the code in the template with the appropriate containers that we created in the previous example. Doing this, we replace the header in the code and tell dotCMS to include the header container instead. We do the same with the navigation, the body and the footer.

After we hit save and publish, our new template is ready to be used to build html pages.

### 1.2.4. Step 3: Creating Folders

Let's click over to the "Browser" tab and take a look at our host. dotCMS allows many different sites and urls to be hosted in a single installation. If we right-click on the host called "localhost", we see that this host will respond and serve pages that use "localhost" as a hostname. It is also set to act as the "default host" which means that if anyone tries to access the server with hostname that cannot be matched, this host will be used to server content as a fall back. Clicking over to the permissions tab, click the checkbox that sets the host to allow for CMS anonymous read access. This will allow any site visitor to view pages without having to authenticate him or herself. Save these changes.

Next things we need to do is to create a few folders. dotCMS tries to enforce good information architecture and does not allow users to create pages in the root directory. Let's create a new folder called "home". We do this by right-clicking on the host icon and selecting "add folder", which brings us to the "add folder" screen. By typing home into the title box, the system will recommend the folder url (dotCMS supports real, search-engine friendly urls by default). Let's also check the "show on menu" box. Checking this box will automatically add the folder to the generated navigation and crumbtrails. Hit save to continue.

Next, let's create a second folder called "images". This folder should be used to only store images, and so we will specify that in the "allowed file extensions" field. This will prevent users from uploading other content into the images folder. After we do this, we save the folder.

Finally, let us create a folder called "services". In the future, this folder will hold a separate section of pages in our website that describes our services. We check the "show on menu" checkbox and hit save.

### 1.2.5. Step 4: Creating a Page

Now we are ready to build our first page. We begin by right-clicking on the "home" folder and selecting "New HTML Page". This brings up the "Add/Edit HTML page" screen.

Let's type our page title "Home Page". Again, the system will automatically suggest a page url. Instead, let's type our own page url, "index". Index is a special page url that indicates that the page is the index page, and is called when a site visitor types a url that only specifies a folder and not a page, like: localhost/home/

We then select the template we created in step 1.3 (this is really all a user has to do to begin creating pages). But before we continue, let's take a look at some other options available to us on a page. Click the "Advanced Properties" tab and you can see that you can set a page to be visible on the menu, set metadata to be entered for a particular page, force a page to be accessible only via over a secure connection and can specify a redirect for any given page. For now, leave everything blank and hit save.
This takes us to the page preview mode.

### 1.2.6. Step 5: Adding content to your page

We can see our new page in our new template and we can see the content controls for our body container. We can now add content to a page. By clicking the down arrow icon, you can select to "add content" to a page. This takes us to the "add content" screen. You will be presented with a content title and a WYSIWYG interface. Enter some content and hit "save and publish". This will return you to the preview mode with your content in place and formatted as laid out in the containers and templates. Hit "Publish Page" and the page will refresh and return as a new published page. Click on "Live Mode".

Congratulations, you have published your first page in dotCMS.  If you have followed these instructions, your page can be publicly viewed by going to the url: [http://localhost/home/](http://localhost/home/)