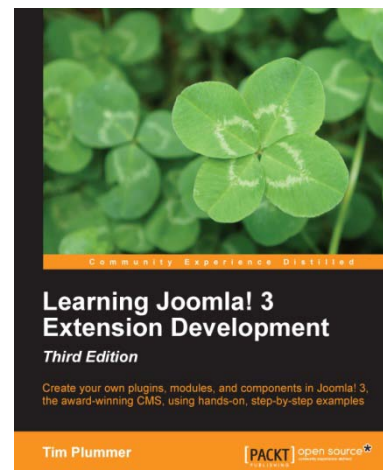


# Learning Joomla! 3 Extension Development

Tim Plummer



## Chapter No.2 " Getting Started with Plugin Development "

## In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.2 " Getting Started with Plugin Development "

A synopsis of the book's content

Information on where to buy this book

## About the Author

**Tim Plummer** is a Joomla! enthusiast and extension developer who has been building websites for over ten years, and specializing in Joomla! since 2008. Tim has developed and maintained several popular components, modules, and plugins, which are listed on the Joomla! Extension Directory, in addition to his day job in IT for a multinational manufacturer. Tim lives in Sydney, Australia, with his wife, Tamlyn, who runs her own design agency and two kids, Zane and Ava-Lily, who keep Tim very busy. At university Tim studied Engineering, and he has a Bachelor of Engineering in Telecommunications Engineering degree, however his passion in IT has been his career focus.

Tim is very active in the Joomla! community; in January 2012 Tim took on the convener role for the Sydney Joomla! User Group (JUG), and he has been coorganizer for the annual Joomla!Day Sydney conference since 2011, taking on the coordinator role in 2013. Tim has run Joomla! development workshops at Joomla! Day conferences and various JUG groups throughout Australia, and is a regular presenter at the Sydney JUG.

**For More Information:**

[www.packtpub.com/learning-joomla-3-extension-development/book](http://www.packtpub.com/learning-joomla-3-extension-development/book)

# Learning Joomla! 3

## Extension Development

This book will give you a step-by-step introduction with practical examples of how to develop plugins, modules, and components for Joomla! 3. It may also be useful for people who wish to make minor modifications to existing components, rather than creating their own extensions from scratch.

Joomla! is one of the world's most popular open source content management systems (CMS), which currently powers approximately 2.7 percent of the websites on the Internet. Joomla! has been downloaded over 35 million times, and has thousands of add-on extensions (apps). Joomla! 3 is the first major CMS to be mobile friendly by default.

Unlike the other two popular open source CMS projects, Joomla! is completely community driven; there is no controlling company or paid staff. Joomla! uses object oriented principles, and is database agnostic. Joomla! is the best mix of functionality, extensibility, and user friendliness.

Developing extensions for Joomla! allows you to harness the full power of Joomla! and build some really great websites and applications. This book is going to give you all the knowledge you need to get started with Joomla! extension development, with lots of practical examples that you can follow along with and learn by doing.

There are many ways that you can get involved with the Joomla! community and contribute to make it better. There is no minimum time commitment; you can contribute as much or as little as you like.

Most major cities have a Joomla! User Group (JUG) that meets regularly to share Joomla! knowledge, which I encourage you to join. Many JUG groups run annual Joomla!Day conferences, which are definitely worth going to, and are a great opportunity to network and pick up some great tips. You can find out about JUG groups and Joomla!Day events at <http://events.joomla.org/>.

Joining the bug squad is a great way for developers to contribute, and it is also a good place to learn more about the Joomla! codebase and to improve your coding skills. You don't even need to contribute code to be part of the bug squad; you can test other people's patches and make sure they work, and at the same time get a better understanding of how it all works. The Joomla! Developer Network site has links to the issue tracker and other information that will help you get started <http://developer.joomla.org/>.

Got a few spare minutes? Jump onto the Joomla! forums and answer a few questions; there are many people who are just starting out with Joomla! and could do with a helping hand <http://forum.joomla.org/>.

**For More Information:**

[www.packtpub.com/learning-joomla-3-extension-development/book](http://www.packtpub.com/learning-joomla-3-extension-development/book)

Developing extensions is a great way to encourage the use of Joomla! and your individual effort can make a big difference. I know many people who made the decision to use Joomla! due to the availability of just one specific third-party extension that solved the problem they had in an efficient and cost-effective way. So I encourage you to get involved full stop I'm sure you'll have fun along the way and make some new friends at the same time.

## What This Book Covers

*Chapter 1, Before you Start*, contains some information that you should know before you start developing Joomla! extensions. It will give you a brief introduction to the extension types in Joomla! as well as highlight some new features of Joomla! 3 compared with previous versions. We will also look at licensing, business models, and coding standards, and touch on upgrading Joomla! 2.5 components to Joomla! 3.

*Chapter 2, Getting Started with Plugin Development*, is where you will create your first plugin for Joomla!. This chapter will introduce some basic concepts such as how the installation XML file works and how to create an installable extension for Joomla!. We will use this plugin later in conjunction with our component.

*Chapter 3, Getting Started with Module Development*, covers module development, both frontend and backend. We will also look at template overrides and alternative layouts, and how to make your module responsive using Bootstrap.

*Chapter 4, Getting Started with Component Development*, covers component development and by the end of this chapter you will have built a very simple component that we will make more complex in the following three chapters. It also explores the numerous JForm field types..

*Chapter 5, Backend Component Development – Part 1*, continues development of our component, but focusing on the backend. Specifically you will learn how to add columns to your view, implement drag and drop ordering, add toolbar buttons and view filters.

*Chapter 6, Backend Component Development – Part 2*, is where you will finish the backend of your component. You will learn about pagination, submenus, ACL, and how to make your component support multiple database types.

*Chapter 7, Frontend Component Development*, covers the development of the frontend of your component. You will learn how to add CSS files, menu item parameters, and how to translate your component. We will also look at how your component can interact with other extensions such as Captcha and a third-party comments component.

*Chapter 8, Security – Avoiding Common Vulnerabilities*, contains some hands-on ethical hacking to teach you about common vulnerabilities and how to avoid them in your extensions.

**For More Information:**

[www.packtpub.com/learning-joomla-3-extension-development/book](http://www.packtpub.com/learning-joomla-3-extension-development/book)

*Chapter 9, Packing Everything Together*, covers how you can prepare your extensions for distribution, as well as managing changes. It also shows how you could set up an update server, and some tips for getting listed on the Joomla! Extension Directory.

*Chapter 10, Extending your Component with Plugins and Modules*, contains a few plugins and modules that extend the functionality of your component, including a smart search plugin. We will also take a look at the new tags feature in Joomla! 3.1 and how you can integrate this into your component.

**For More Information:**

[www.packtpub.com/learning-joomla-3-extension-development/book](http://www.packtpub.com/learning-joomla-3-extension-development/book)

# 2

## Getting Started with Plugin Development

By now, I'm sure you are itching to get your hands dirty and start playing with some code. If you skipped the first chapter, don't worry, you can come back to the theory later. Most development books will start out with a "hello world" example, but that's been done before and is really boring, so we are going to skip that and start out by creating a simple plugin that actually does something useful. By the end of this chapter, you will have made your first Joomla! plugin. You will learn the following things in this chapter:

- An overview of the various types of Joomla! plugins
- How to create a content plugin
- How to package your plugin so you can share it with others
- How to make your plugin more flexible by adding parameters
- How to create language files for your plugin
- The power of language overrides

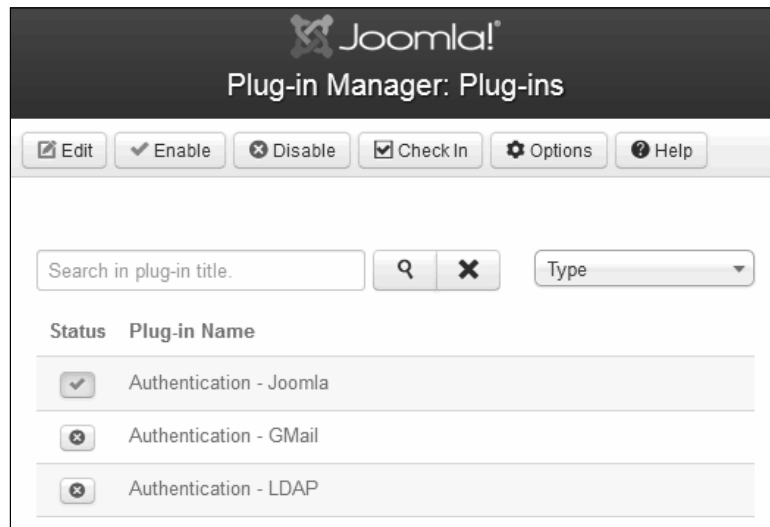
**For More Information:**

[www.packtpub.com/learning-joomla-3-extension-development/book](http://www.packtpub.com/learning-joomla-3-extension-development/book)

## Plugin types

Before we create our first plugin, let's talk about the different types of plugins that you can create for Joomla!. Plugin code is executed when certain events are triggered. You can skip the next couple of pages if you just want to get stuck in with the hands-on example and just go straight to the *Where do I start?* section, but you may want to refer back to this introduction later when you are creating your own plugins. The different types of plugins are explained as follows:

- **Authentication:** When you log in to your website, an authentication plugin runs to check your credentials and decide if you should have access to the site. Authentication plugins are used when you want to use other login methods too, such as **Facebook**, **Gmail**, or **LDAP**. You can use more than one authentication plugin on the same site so that you can give your users a choice and not force them to have to remember another password by allowing them to reuse one of their existing accounts in another system. The following screenshot shows the authentication plugins you can enable on your Joomla! site:



### Downloading the example code

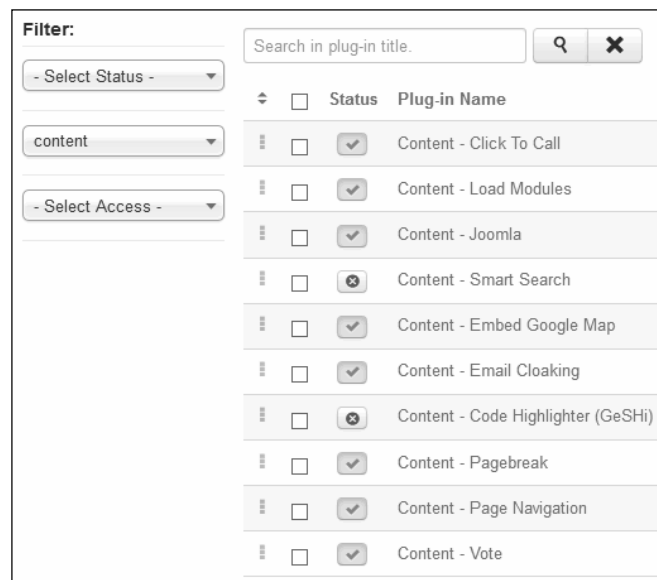


You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

- **Captcha:** You have probably seen a CAPTCHA on a form that you've filled out. It's that box where you need to type in characters to prove that you are a person and not an automated spambot. Joomla! has a plugin in the core for reCAPTCHA, but potentially other CAPTCHAs could be added if you wrote an appropriate plugin. The following screenshot shows a Captcha where you need to type in the given Captcha characters to prove you are not an automated spambot:

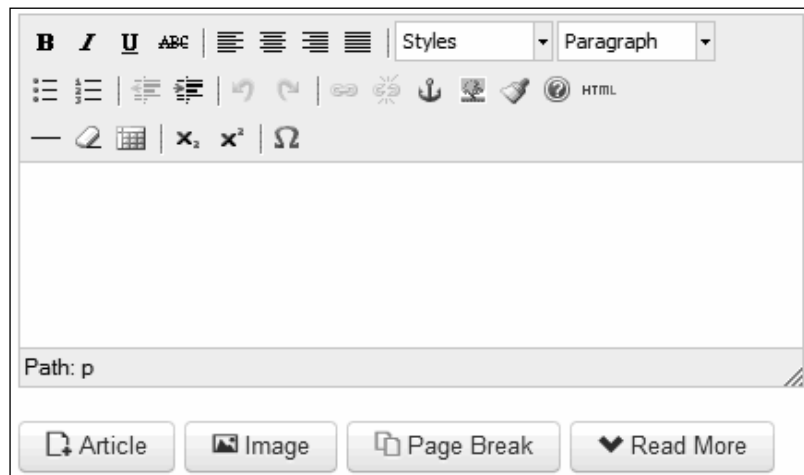


- **Content:** This mostly refers to article contents but some extensions load content plugins too. Content plugins allow you to change the content before it is displayed on the website. The **Click To Call** plugin we are creating in this chapter is a content plugin. We can see the **Click to Call** plugin enabled in the following screenshot:

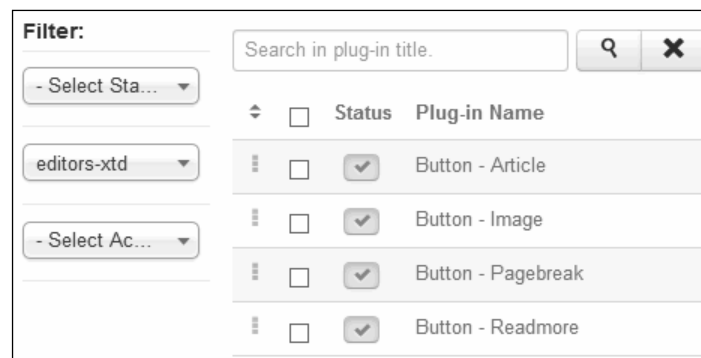




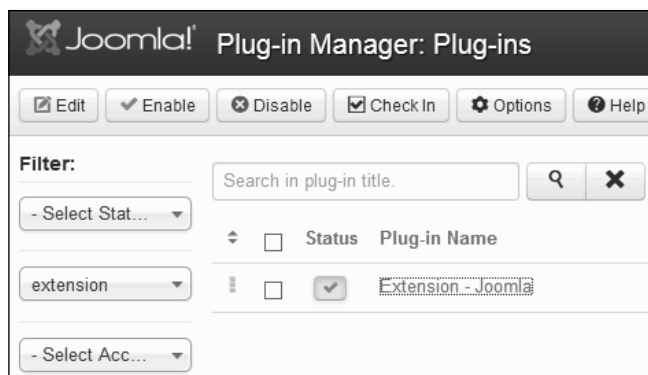
- **Editors:** Joomla! has **TinyMCE** and **CodeMirror** editors in the core, but many people want a bit more functionality from their editors and install another third-party editor such as JCE. You could potentially integrate a different editor in the core by creating an editor plugin. Editor plugins are probably the most complex, so it's not something a beginner would attempt to create.



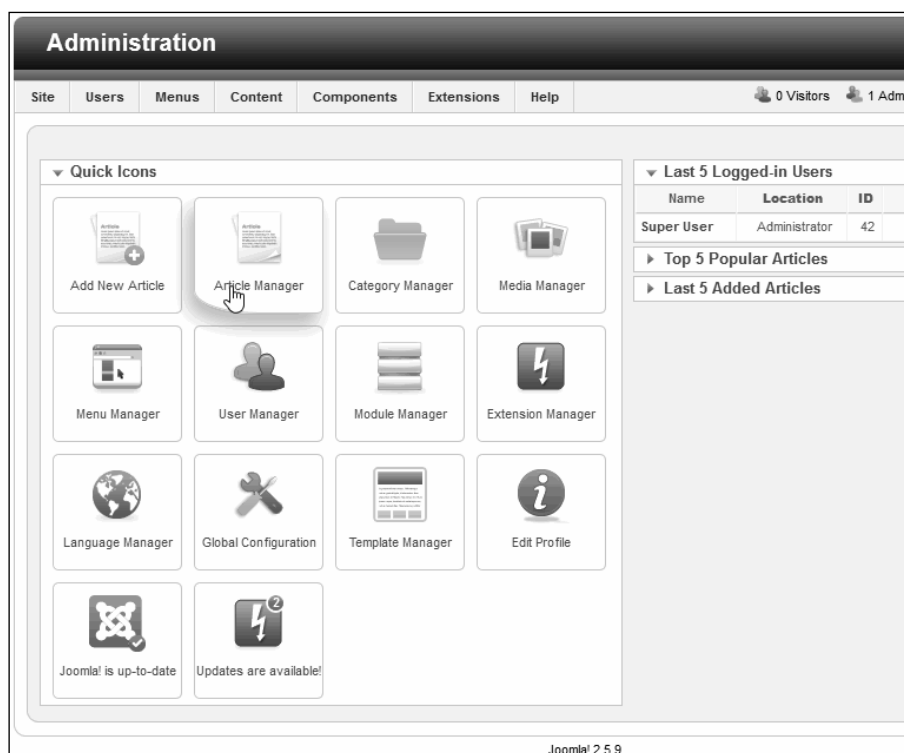
- **Editors-XTD:** This type of plugin is to extend the functionality of editor plugins by adding new buttons which trigger new functionality. For example, if you wanted to create a button in the editor that inserts a copyright symbol, you could write an editors-XTD plugin with this functionality.



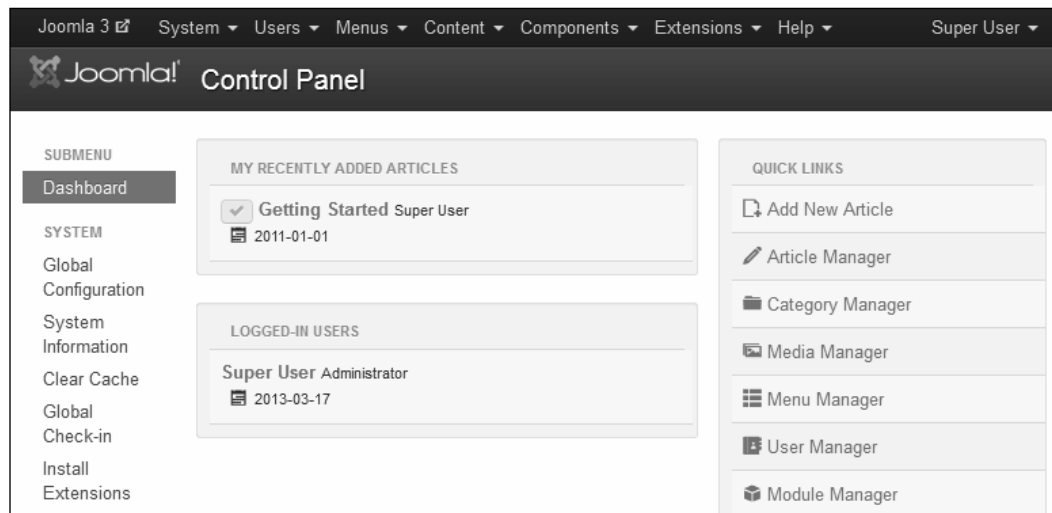
- **Extension:** Extension events are triggered when extensions are installed, uninstalled, updated, or edited through Plug-in Manager, Module Manager, Template Manager, or Language Manager. I haven't seen this plugin type used much in third-party extensions.



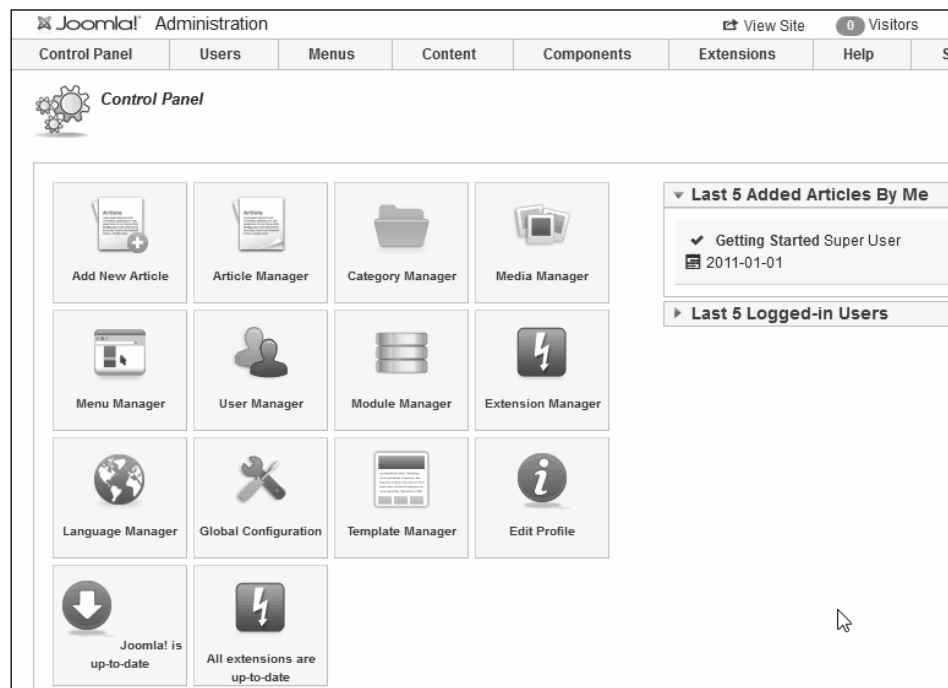
- Quick Icons:** In the admin control panel page in the backend of Joomla! versions prior to Joomla! 3, there was a Quick Icons panel that provided shortcuts to many of the frequently accessed sections. The Joomla! 3 core still has full support for Quick Icons, however the default administrator template is lacking this feature. **Quick Icons** plugins are used to display the Joomla! update icons, which actually change to give a visual indication that there is an update available.



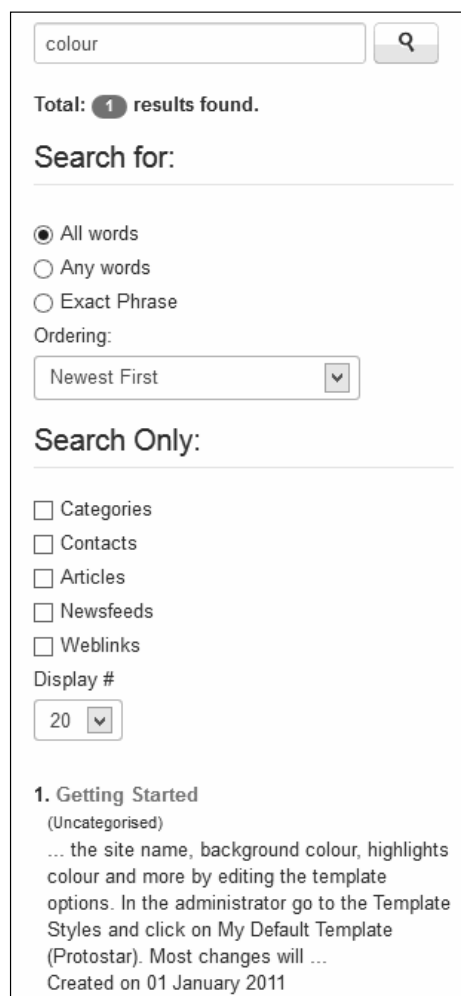
This Quick Icons panel is not shown in the new default template in Joomla! 3 called Isis.



However at the time of writing this book, the alternate admin template in Joomla! (called Hathor) does still include these icons.



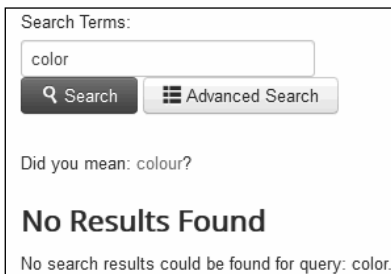
- **Search:** This is the older type of search that has been in Joomla! for a while now. The search plugin type is commonly used by extension developers to get data from their component showing up in the core Joomla! search. Due to the introduction of smart search, eventually this older style search will be removed from the core, but for now, some people still choose to use it.



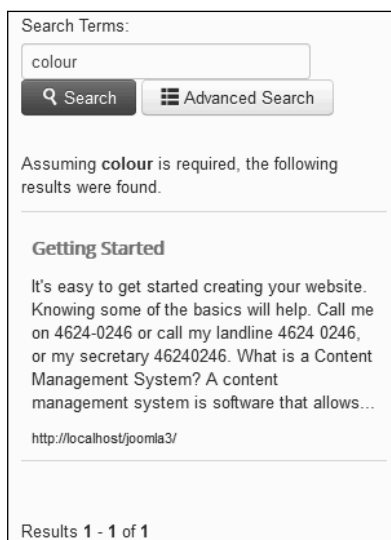
The screenshot displays the Joomla! search interface. At the top, a search bar contains the text 'colour' and a magnifying glass icon. Below the search bar, it states 'Total: 1 results found.' followed by 'Search for:'. There are three radio button options for search criteria: 'All words' (selected), 'Any words', and 'Exact Phrase'. Below these is an 'Ordering:' dropdown menu set to 'Newest First'. Underneath is a 'Search Only:' section with five checkboxes: 'Categories', 'Contacts', 'Articles', 'Newsfeeds', and 'Weblinks', all of which are currently unchecked. Below the checkboxes is a 'Display #' dropdown menu set to '20'. The search results section is titled '1. Getting Started' and includes the text '(Uncategorised)', '... the site name, background colour, highlights colour and more by editing the template options. In the administrator go to the Template Styles and click on My Default Template (Protostar). Most changes will ...', and 'Created on 01 January 2011'.

- **Smart Search (Finder):** Smart Search is the new search functionality that was added to the Joomla! core in Joomla! 2.5. It does some fancy stuff such as suggesting alternative keywords that might help the user find what they are looking for. For example, if you search for "color" (American spelling) it will suggest "colour" (UK spelling), or if you made a typo, it would suggest similar words. Because this new search was such a dramatic change, it was decided that it would be gradually introduced by having it as an optional feature that you can turn on if you want to. Many extension developers have written smart search plugins for their components so they can take advantage of this new searching ability. These plugins are used to index the content so it can be used by Smart Search.

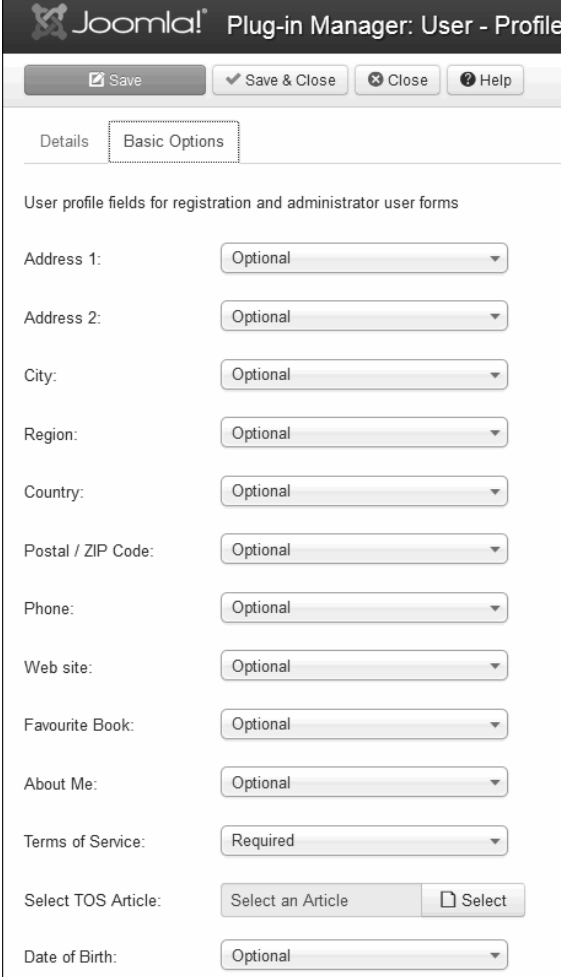
The following screenshot demonstrates how smart search can suggest similar words to the term you are searching:



The following screenshot shows an example of a search using Smart Search where a relevant article is found.

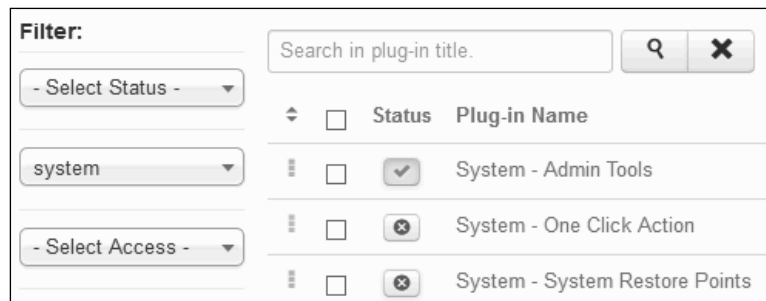


- **User:** User plugins allow you to do some extra things during user-specific events. The user profile plugin is a good example that was added in Joomla! 2.5 which extends the fields on the user registration form and allows you to capture additional details such as address, terms of service, date of birth, and website. If you wanted to keep a log of when people log in and log out of your website, you could write a user plugin to do this.



The screenshot displays the Joomla! Plug-in Manager interface for the 'User - Profile' plugin. The interface has a dark header with the Joomla! logo and the title 'Plug-in Manager: User - Profile'. Below the header is a toolbar with buttons for 'Save', 'Save & Close', 'Close', and 'Help'. The main content area has two tabs: 'Details' and 'Basic Options', with 'Basic Options' being the active tab. Under the 'Basic Options' tab, there is a section titled 'User profile fields for registration and administrator user forms'. This section contains a list of fields, each with a label and a dropdown menu. The fields are: Address 1, Address 2, City, Region, Country, Postal / ZIP Code, Phone, Web site, Favourite Book, About Me, Terms of Service, Select TOS Article, and Date of Birth. Most fields have a dropdown menu set to 'Optional', while 'Terms of Service' is set to 'Required'. The 'Select TOS Article' field has a dropdown menu set to 'Select an Article' and a 'Select' button next to it.

- **System:** System plugins run in the background and are triggered during every execution cycle regardless of which task is being performed. Akeeba Backup uses system plugins to trigger the update e-mail notifications. System plugins are loaded very early in the Joomla! execution cycle, before most Joomla! core classes, so we can use system plugins to override the core classes and change how Joomla! works. This is because when Joomla! loads a class into the memory, it first checks to see if this class is already loaded into the working memory, and if so, it won't load it again. So if our system plugin gets in first, its code will be used instead of the core class. Overriding core Joomla! classes is an advanced topic, and is outside the scope for this book. The following screenshot shows the System plugins which you can enable/disable as needed:



## Plugin event triggers

Each plugin type has its own set of events that trigger these plugins to execute. Each plugin can use one or more event triggers but there is no need to use all the event triggers available, just pick out the ones you need. You also have the freedom to create your own custom plugin event triggers in your components, however, that is also outside the scope of this book. The following table summarizes the types of plugins available and their respective event triggers:

Plugin type	Event trigger
Authentication	onUserAuthenticate
	onUserAuthorisationFailure
Captcha	onCheckAnswer
	onDisplay
	onInit

Plugin type	Event trigger
Content	onContentAfterDelete
	onContentAfterDisplay
	onContentAfterSave
	onContentAfterTitle
	onContentBeforeDelete
	onContentBeforeDisplay
	onContentBeforeSave
	onContentChangeState
	onContentPrepare
	onContentPrepareData
	onContentPrepareForm
	onContentSearch
	onContentSearchAreas
	onDisplay
Editors	onGetContent
	onGetInsertMethod
	onInit
	onSave
	onSetContent
Editors-XTD	onDisplay
Extension	onExtensionAfterInstall
	onExtensionAfterSave
	onExtensionAfterUninstall
	onExtensionAfterUpdate
	onExtensionBeforeInstall
	onExtensionBeforeSave
	onExtensionBeforeUninstall
	onExtensionBeforeUpdate
Quick Icons	onGetIcons
Search	onContentSearchAreas
	onContentSearch



Plugin type	Event trigger
Smart Search	onFinderAfterDelete
	onFinderAfterSave
	onFinderBeforeSave
	onFinderCategoryChangeState
	onFinderChangeState
System	onAfterDispatch
	onAfterInitialise
	onAfterRender
	onAfterRoute
User	onUserAfterDelete
	onUserAfterSave
	onUserBeforeDelete
	onUserBeforeSave
	onUserLogin
	onUserLogout

## Where do I start?

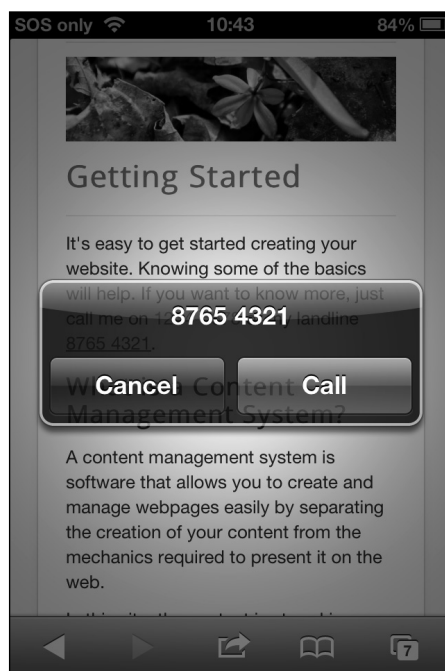
To get started with plugin development, first we need to decide what problem we are trying to solve or what functionality we are trying to implement.

Let's assume that you have a website with various phone numbers listed on the site, and that this site is popular with people who access the site through a smartphone. Currently, to call one of the numbers, the user has to remember the number to dial it, or perhaps they jot it down on a piece of paper or attempt to select and copy the number through their browser. This is not very user-friendly, a much better way to do this would be to change the phone number to a **click-to-call link**, so the user can just click the phone number and their phone will call it. Now, you could go through the website and manually change all the phone numbers to these links, but that would be inconvenient. Assuming our phone number is in the format 1234-5678, we could change it to this HTML code to create a click-to-call link.

```
<a href="tel: 1234-5678">1234-5678</a>
```

Wouldn't it be better if we wrote a plugin that identified phone numbers in all our Joomla! articles and automatically changed them to these click-to-call links? Well that's exactly what we are going to do.

The following screenshot is an example of what the plugin will do when we have finished creating it. When a mobile user clicks on the phone number, it will allow them to easily call the number.



So now that we know what we are trying to achieve, the first thing to do would be to check the **Joomla! Extension Directory (JED)** to make sure that a solution to this problem does not already exist. Why reinvent the wheel if you can find something that already does the job or does something similar you could adapt? So let's assume that we've had a look at the JED (<http://extensions.joomla.org>) and we can't find a suitable plugin and we want to create our own.

The next step is to come up with a unique name for this extension; in this case we will call it `plg_content_clicktocall`. The first part of the name (`plg`) indicates that this is a plugin extension. Then we identify what type of plugin this is; in this case, it's a content plugin. Then we add our unique name, preferably something short that describes this plugin; in our case, this is `clicktocall`. Once you have a name, it's a good idea to search the JED and Google to make sure that it is truly unique as you will have problems listing the extension on the JED later if it is not unique. Many developers add their initials or business name to the extension to avoid name conflicts.

## Creating the installation XML file

The first file we need to create is the installation XML file (also known as the XML Manifest) which tells Joomla! all about this plugin when you install it through an extension manager. This XML file describes all the files and folders used by this plugin, and what parameters the plugin has. The naming standard of the installation XML file is `extensionname.xml`, in our case it will be `clicktocall.xml` as we drop the `plg_content_` prefix.

Create a folder called `plg_content_clicktocall`.

In this folder, create a file called `clicktocall.xml` and insert the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<extension
  version="3.0"
  type="plugin"
  group="content"
  method="upgrade">
  <name>Content - Click To Call</name>
  <author>Tim Plummer</author>
  <creationDate>April 2013</creationDate>
  <copyright>Copyright (C) 2013 Packt Publishing. All rights
reserved.</copyright>
  <license> http://www.gnu.org/licenses/gpl-3.0.html</license>
  <authorEmail>example@packtpub.com</authorEmail>
  <authorUrl>http://packtpub.com</authorUrl>
  <version>1.0.0</version>
  <description>This plugin will replace phone numbers with click
to call links. Requires Joomla! 3.0 or greater.
  Don't forget to publish this plugin!
  </description>
  <files>
    <filename plugin="clicktocall">clicktocall.php</filename>
    <filename>index.html</filename>
  </files>
</extension>
```

Now let's examine what we have just made. The first thing to notice is the extension tag, that tells Joomla! what kind of extension this is; in our case, a plugin.

```
<extension
  version="3.0"
  type="plugin"
  group="content"
  method="upgrade">
```

The version indicates the minimum version of Joomla! that this extension can be installed on. If you wanted to also support Joomla! 2.5, you could to change this to `version="2.5"`.

```
<extension
  version="2.5"
  type="plugin"
  group="content"
  method="upgrade">
```

The group attribute is required for plugins because there are several types of plugins, but you wouldn't see this in a module or a component. In our case, we are creating a content plugin.

```
<extension
  version="3.0"
  type="plugin"
  group="content"
  method="upgrade">
```

The `method="upgrade"` is important to note. If you do not include this, then every time you install an upgrade version of the extension, the uninstall script will run and it will delete your existing data. That's not really a problem for this plugin as it doesn't have an associated database table, but it's a good idea to include it anyway in all your extensions.

```
<extension
  version="3.0"
  type="plugin"
  group="content"
  method="upgrade">
```

In the name tag, you will see `Content - Click To Call`. The standard for plugins is to say the plugin type dash plugin name.

```
<name>Content - Click To Call</name>
```

The `author`, `creationDate`, `copyright`, `license`, `authorEmail`, and `authorUrl` tags are pretty self-explanatory, just put in your own details as follows:

```
<author>Tim Plummer</author>
<creationDate>April 2013</creationDate>
<copyright>Copyright (C) 2013 Packt Publishing. All rights
reserved.</copyright>
<license> http://www.gnu.org/licenses/gpl-3.0.html</license>
<authorEmail>example@packtpub.com</authorEmail>
<authorUrl>http://packtpub.com</authorUrl>
```

The version tag is important, each time you release a new version of your extension you should adjust this version number. You should use a three digit version number separated by full stops.

```
<version>1.0.0</version>
```

The description tag is the text that the user will see when they initially install the plugin. Most plugin developers add the words "Don't forget to publish this plugin!" as plugins are disabled by default when they are installed and will not work until they have been enabled.

```
<description>This plugin will replace phone numbers with Click to  
Call links. Requires Joomla! 3.0 or greater.  
Don't forget to publish this plugin!  
</description>
```

The files tag tells Joomla! all the files that this extension uses, in this case, we only have two files in addition to our installation XML file, the `index.html` file and the `clicktocall.php` file which is going to contain the code for our plugin.

```
<files>  
  <filename plugin="clicktocall">clicktocall.php</filename>  
  <filename>index.html</filename>  
</files>
```



In the older Joomla! versions, you had to individually list every file in the extension, however, now you can use the `folder` tag to include all the files and subfolders within a folder. For example, if we had a folder called `views`, you could write `<folder>views</folder>`.

Then at the end of our XML file, we have a closing extension tag that tells Joomla! that it's the end of this file.

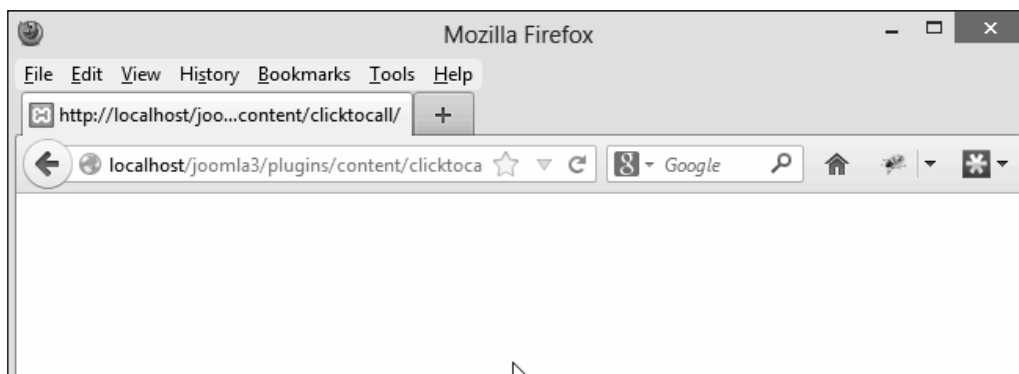
```
</extension>
```

Now in your `plg_content_clicktocall` folder, create an `index.html` file with the following code:

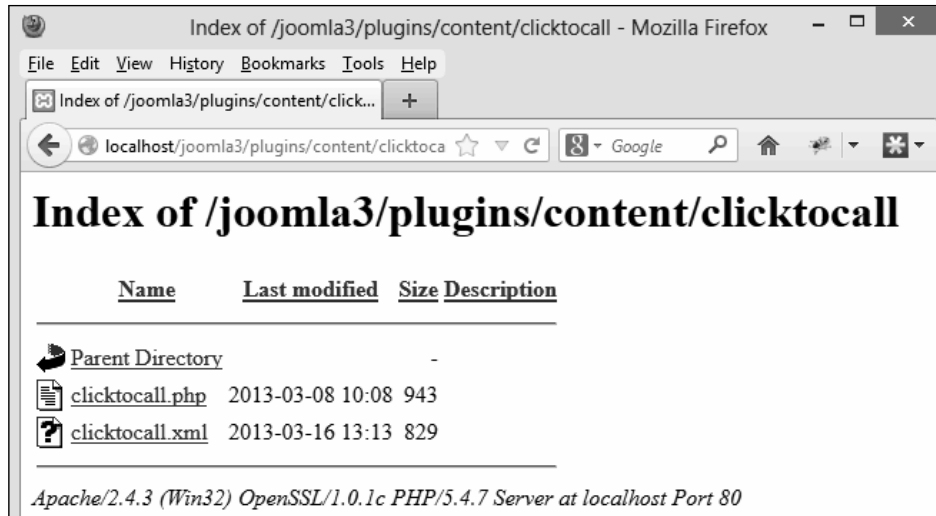
```
<html><body bgcolor="#FFFFFF"></body></html>
```

The `index.html` file should be included in every folder within your extension and will just show the user a blank page if they try to browse the folder directly. In the past, it was considered a security risk not to include these `index.html` files, as without them you could potentially be exposing information about your website to malicious users. The JED previously rejected any extension that did not include these `index.html` files. As of April 1, 2013, the JED has dropped the requirement to include `index.html` files in all folders. Nicholas K. Dionysopoulos from Akeeba has written a detailed explanation as to why the `index.html` files are not an effective security feature. This explanation can be found at <http://www.dionysopoulos.me/blog/86-the-files-of-wrath.html>.

Ideally your web host should have the directory traversal disabled, or you should utilize the `.htaccess` or `web.config` file to prevent directory traversal. However, there are still a lot of Joomla! sites that will not configure these alternatives, so I would be inclined to still include an `index.html` file in every folder. The following screenshot shows a blank page displayed when the user tries to browse a folder directly:



If we were to forget to put in the `index.html` file and the directory traversal was not disabled, someone could potentially browse directly to the folder containing our plugin and they would see all the files in that directory as well as other information such as web server and PHP versions. This is shown in the following screenshot:



## Creating the plugin PHP file

So now we need to write the PHP file that does all the hard work for this plugin and implements the functionality that we are trying to achieve. In your `plg_content_clicktocall` folder, create the file `clicktocall.php` with the following code:

```
<?php
defined('_JEXEC') or die;

jimport('joomla.plugin.plugin');

class plgContentClicktocall extends JPlugin
{
    function plgContentClicktocall( &$subject, $params )
    {
        parent::__construct( $subject, $params );
    }

    public function onContentPrepare($context, &$row, &$params,
$page = 0)
```

---

```

    {
        // Do not run this plugin when the content is being indexed
        if ($context == 'com_finder.indexer')
        {
            return true;
        }

        if (is_object($row))
        {
            return $this->clickToCall($row->text, $params);
        }
        return $this->clickToCall($row, $params);
    }

    protected function clickToCall(&$text, &$params)
    {
        // matches 4 numbers followed by an optional hyphen or space,
        // then followed by 4 numbers.
        // phone number is in the form XXXX-XXXX or XXXX XXXX
        $pattern = '/(\W[0-9]{4})-? ?(\W[0-9]{4})/';

        $replacement = '<a href="tel:$1$2">$1$2</a>';
        $text = preg_replace($pattern, $replacement, $text);

        return true;
    }
}

```

Normally you would expect to see DocBlocks before each class and function, but I've excluded these in this book just to simplify the code. You'll notice that all of the core code has these DocBlock comments which makes it easy for automated tools to generate documentation of APIs, and it also helps some IDEs to provide code completion. In third-party extensions, they are optional but commonly used, and it is recommended that you include them. Here is an example of a typical DocBlock.

```

/**
 * ClickToCall Content Plugin
 *
 * @package      Joomla.Plugin
 * @subpackage   Content.clicktocall
 * @since        3.0
 */

```



Now if we examine the code in the PHP file, you'll see the `defined('_JEXEC')` or `die` which should appear at the top of every PHP file in your extension. It is a security risk if you leave these out as they prevent the PHP file from being executed directly and force it to only run when called by Joomla!. In the past it was common to write a message with the `die` statement, for example, `die( 'Restricted access' )`, but now it is recommended to add just `die` to a blank page and give the potential hacker as little information as possible. A hacker seeing the text **Restricted access** might be able to determine that the website is using Joomla! 1.5, whereas a blank page does not reveal anything.

The next line of code you will see is the `jimport` call that loads the standard Joomla! plugin classes from the core.

```
jimport('joomla.plugin.plugin');
```

We then extend the standard `JPlugin` class to create our own unique class for our plugin.

```
class plgContentClicktocall extends JPlugin
```

When an instance of this class is created, the `__construct` function is called to set it all up, and in our case, we are just loading the `__construct` function from the parent class in the Joomla! core. If we wanted to add in our own custom code, we could add it before or after the `parent::__construct` line.

```
function plgContentClicktocall( &$subject, $params )
{
    parent::__construct( $subject, $params );
}
```

Now you will see a function called `onContentPrepare`, which is one of the plugin events we mentioned earlier. When Joomla! is preparing an article content, it will run this plugin code prior to displaying the output on the page, so it gives us the opportunity to change this content so that what's shown in the browser is different to what's actually stored in the database. When the smart search is not running, we call the `clickToCall` function that does all the hard work of this plugin. When the smart search is indexing content, we want to index the original article content which doesn't need Click to Call links, so there is no need for this plugin to run. As this is a simple plugin, the code from the `clickToCall` function could have been put within `onContentPrepare`, but in most cases, you are trying to do something a bit more complex, so it's better to separate it.

```
public function onContentPrepare($context, &$row, &$params, $page
= 0)
{
    // Do not run this plugin when the content is being indexed
```

```

if ($context == 'com_finder.indexer')
{
    return true;
}

if (is_object($row))
{
    return $this->clickToCall($row->text, $params);
}
return $this->clickToCall($row, $params);
}

```

Now we finally come to the code that identifies a phone number in the article contents and replaces it with our Click to Call link. We are using a **regular expression pattern (regex)** to identify the phone number in the article. Regex code can be quite complex and difficult to understand, which is why I've added lots of comments to explain the purpose of regex, which will make it easier if someone wanted to modify this later, for example, to include an area code in the phone number identification. Once the phone number has been identified, it is replaced using a standard PHP function `preg_replace`.

```

protected function clickToCall(&$text, &$params)
{
    // matches 4 numbers followed by an optional hyphen or space,
    // then followed by 4 numbers.
    // phone number is in the form XXXX-XXXX or XXXX XXXX
    $pattern = '/(\W[0-9]{4})-? ?(\W[0-9]{4})/';

    $replacement = '<a href="tel:$1$2">$1$2</a>';
    $text = preg_replace($pattern, $replacement, $text);

    return true;
}

```

---

### Regex cheat sheet

Regular expression	What it matches to
[a-z]	This expression matches to any lowercase letter.
[^A-Z]	This expression matches to any character that is not a uppercase letter.
[a-z]+	This expression matches to one or more lowercase letters.
[abc]	This expression matches to any of the characters a, b, or c.
[^A-Za-z0-9]	This expression matches to any symbol (not a number or a letter).

---

### For More Information:

[www.packtpub.com/learning-joomla-3-extension-development/book](http://www.packtpub.com/learning-joomla-3-extension-development/book)

#### Regex cheat sheet

Regular expression	What it matches to
[0-9]	This expression matches to any number digit.
[0-9]{3}	This expression matches to any three digit number.
\W	This expression matches to any non "word" character.
\w	This expression matches to any "word" character.

## Zip it up and install your plugin

Now that you have created all the code for the Click to Call plugin, you need to test it in your Joomla! site to see if it works correctly. We will start out with a way that you might get this running on your development site, then, we will zip it all up into an installable file that you could distribute to other people.

Before we do that, you will need to add some phone numbers in your articles that we are going to turn into Click to Call links, so that we know the plugin works. Add some phone numbers in 1234-5678 or 1234 5768 format.

Title \*

Category \*  

- Uncategorised

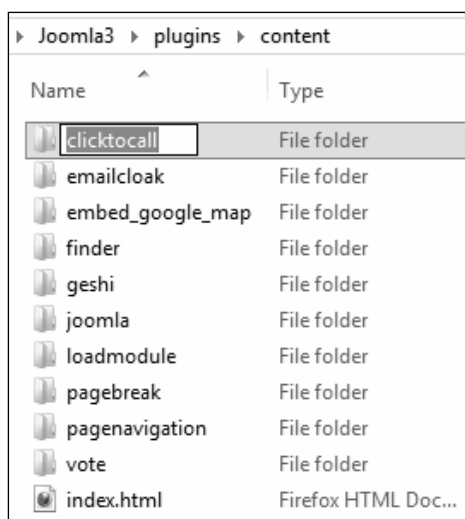
**B** *I* U ABC | | | HTML

| x<sub>2</sub> x<sup>2</sup> | Ω

It's easy to get started creating your website. Knowing some of the basics will help. If you want to know more, just call me on 1234-5678 or my landline 8765 4321.

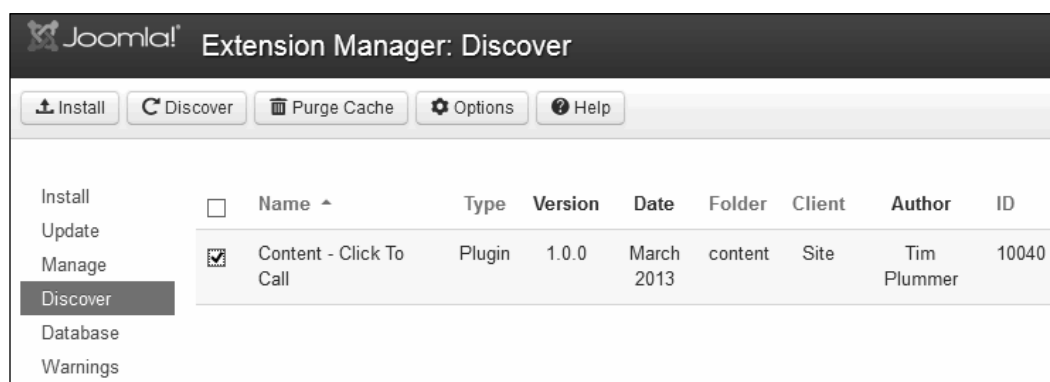
What is a Content Management System?

In your Joomla! 3 development website, find the `/plugins/content/` folder and create a new folder `clicktocall`.

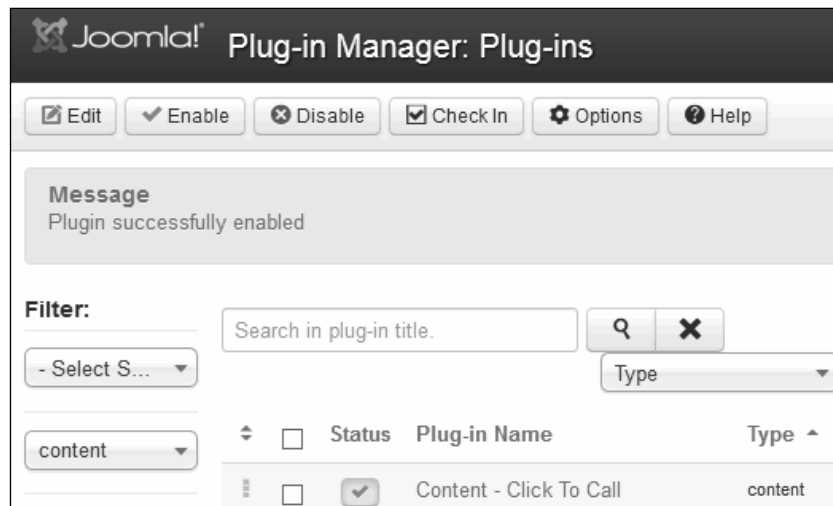


Copy into this folder the files you have created, `clicktocall.xml`, `clicktocall.php`, and `index.html`.

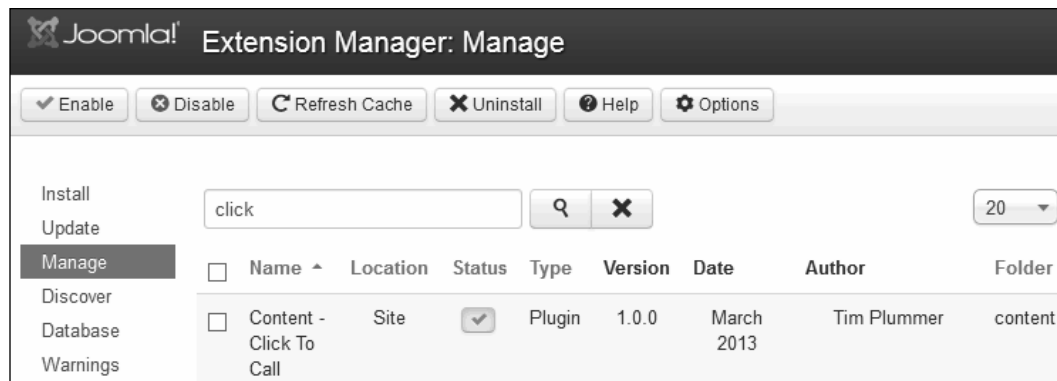
Now in your Joomla! Administrator, navigate to **Extensions | Extension Manager**, and select the **Discover** view at the left side. Click on the **Discover** button and you should see the **Content - Click To Call** plugin appearing. Select it and click on the **Install** button.




Now you will need to enable this plugin through **Extensions | Plug-in Manager**. Just click on the red cross in the status column and it should turn into a green tick.

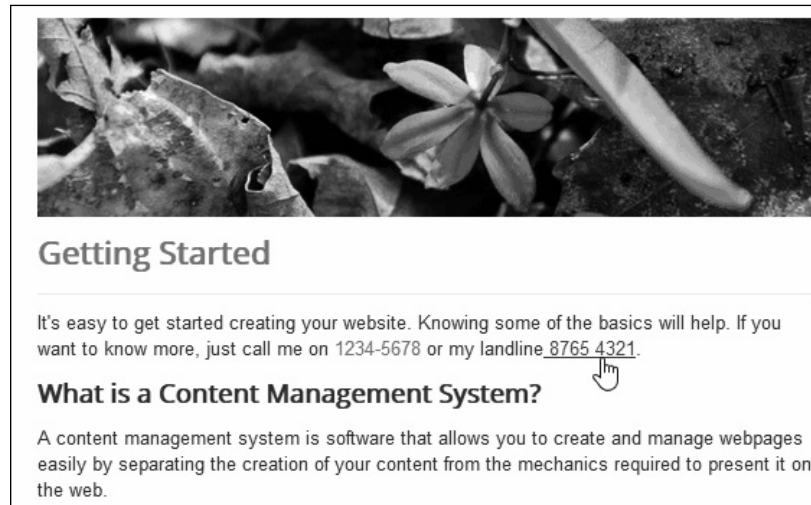


You will also notice that your plugin is listed by navigating to **Extensions | Extension Manager | Manage**. You can also see the information such as **Date**, **Author**, and **Version** that you set in your installation XML file are shown here.

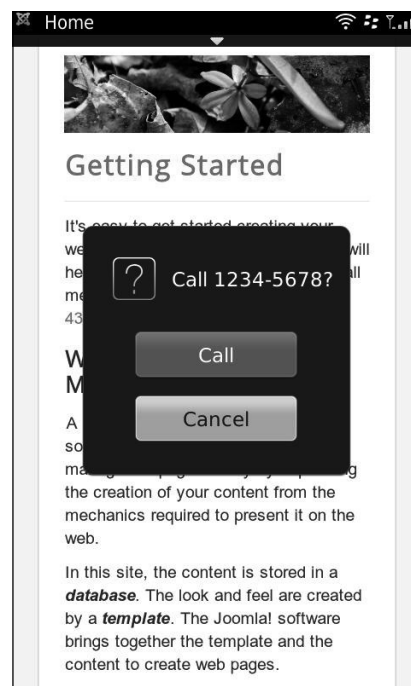


 Plugins are executed in the order in which they appear when you click on the ordering column. When you have more than one plugin triggering for an event, the results of one plugin can affect the next, so keep that in mind.

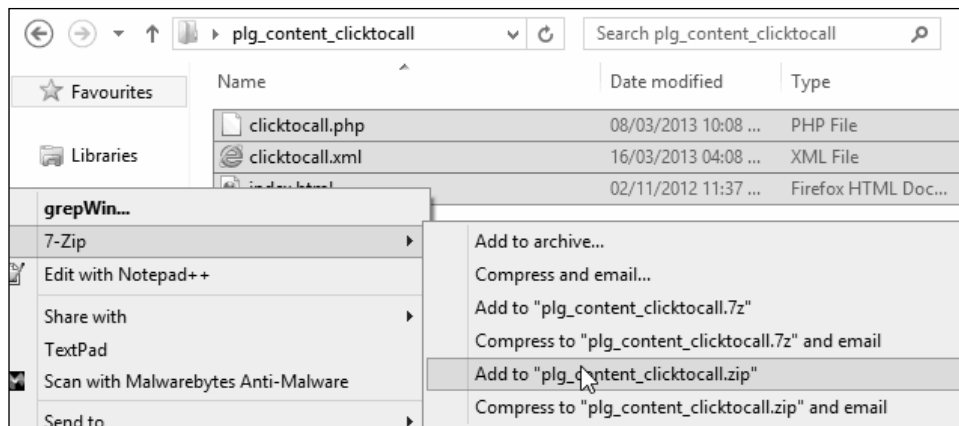
Take a look at your frontend and you should now see all the phone numbers you inserted earlier have turned into Click to Call links.



If you view this on a mobile device, you will see how the Click to Call link works.



Now that we know the plugin works, you will want to be able to package this up into a file that you can give to other people to install on their Joomla! websites. Well, the good news is that you have already done all the hard work, all you need to do is chuck these files you created into the same folder and zip it up. You can use whatever zip program you like, for example 7-zip. Joomla! supports both .zip and .tar.gz formats, so you can choose which one to use. Generally the .tar.gz files are smaller; however some web hosts don't support .tar.gz files, in which case you could send those users a .zip file instead. The .zip file format is most commonly used by extension developers.



You should call your plugin file `plg_content_clicktocall_v1.0.0.zip`, so just by looking at the filename, someone could tell that this is a content plugin called Click to Call and it's Version 1.0.0. This ZIP file can now be installed via the normal extension manager installer.

When testing your extensions in Joomla!, you should make sure that you set **error\_reporting** to **maximum** in your global configuration. When error reporting is set lower such as the default setting, a lot of warning messages, notices, and error messages are hidden from the browser. It is best to see these messages so you can address the problem and improve your code before your users start complaining.

The screenshot shows the 'Server Settings' dialog box in Joomla!. It contains four settings: 'Path to Temp Folder' with a text input field containing 'C:\xampp\htdocs\packt/tmp'; 'Gzip Page Compression' with a toggle switch set to 'No'; 'Error Reporting' with a dropdown menu set to 'Maximum' (highlighted with a red rectangle); and 'Force SSL' with a dropdown menu set to 'None'.

You may also need to adjust the **error\_reporting** setting in your `php.ini` file on your development site.

```
error_reporting = E_ALL
```

## Adding the parameters to our plugin

Our plugin works well, but what if the phone number format in your country is not 1234-5678? For example, Germany uses the format 123-45678. Wouldn't it be better if we had a parameter where we could set the phone number format? As this is a simple plugin, we are not going to worry about area code prefix or numbers in more complex formats such as 123-456-7890; we are concentrating on numbers that consist of two groups of digits.

We are going to add two parameters to our plugin (also known as options), one that sets the number of digits in the first group, and one that sets the number of digits in the second group.

So open up your `clicktocall.xml` file, and add in the highlighted code shown as follows (everything between the config tags):

```
<?xml version="1.0" encoding="UTF-8"?>
<extension
  version="3.0"
  type="plugin"
  group="content"
  method="upgrade">
  <name>Content - Click To Call</name>
  <author>Tim Plummer</author>
  <creationDate>April 2013</creationDate>
```



```
<copyright>Copyright (C) 2013 Packt Publishing. All rights
reserved.</copyright>
<license>http://www.gnu.org/licenses/gpl-3.0.html</license>
<authorEmail>example@packtpub.com</authorEmail>
<authorUrl>http://packtpub.com</authorUrl>
<version>1.1.0</version>
<description>This plugin will replace phone numbers with Click
to Call links. Requires Joomla! 3.0 or greater.
Don't forget to publish this plugin!
</description>
<files>
  <filename plugin="clicktocall">clicktocall.php</filename>
  <filename>index.html</filename>
</files>

<config>
  <fields name="params">
    <fieldset name="basic">

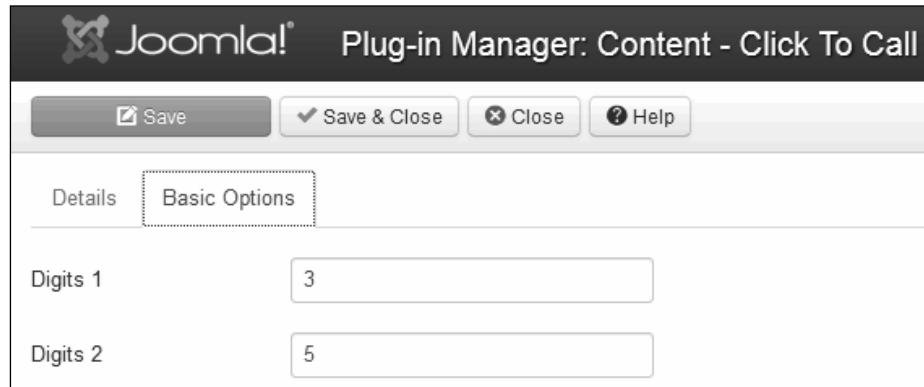
      <field name="phoneDigits1" type="text"
        default="4"
        label="Digits 1"
        description="How many digits in the first part of the
phone number?"
      />
      <field name="phoneDigits2" type="text"
        default="4"
        label="Digits 2"
        description="How many digits in the second part of the
phone number?"
      />

    </fieldset>
  </fields>
</config>
</extension>
```

Besides incrementing the version number, we have added two text fields that default to 4 which was our original number of digits.

```
<field name="phoneDigits1" type="text"
  default="4"
  label="Digits 1"
  description="How many digits in the first part of the phone
number?"
/>
```

If you take a look in your plugin through the plugin manager, you will see these new parameters added on a **Basic Options** tab. But we still need to edit our PHP file to actually use these parameters.



Open up your `clicktocall.php` file, and make the following changes to your `clickToCall` function.

```
protected function clickToCall(&$text, &$params)
{
    $phoneDigits1      = $this->params->get('phoneDigits1', 4);
    $phoneDigits2      = $this->params->get('phoneDigits2', 4);

    // matches 4 numbers followed by an optional hyphen or space,
    // then followed by 4 numbers.
    // phone number is in the form XXXX-XXXX or XXXX XXXX
    $pattern = '/(\W[0-9]{'. $phoneDigits1. '})-? ?(\W[0-9]{'. $phoneDigits2. '})/';

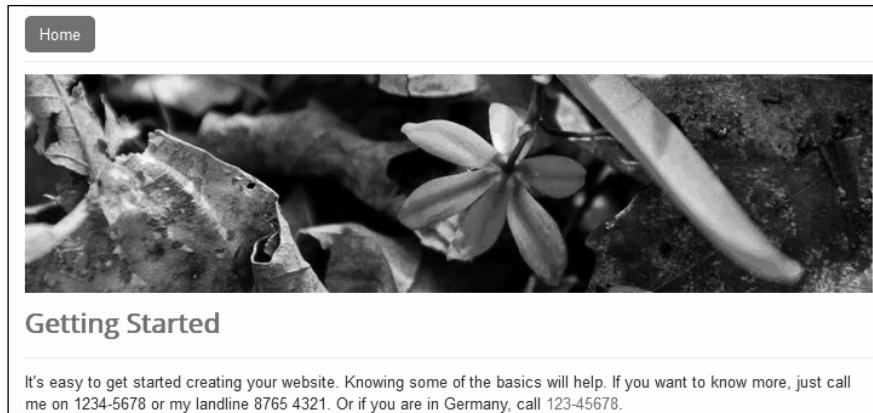
    $replacement = '<a href="tel:$1$2">$1$2</a>';
    $text = preg_replace($pattern, $replacement, $text);

    return true;
}
```

We are loading up the `phoneDigits1` parameter and putting it into a `$phoneDigits1` variable which we use in the pattern line. Notice that we are loading a default value of 4 just in case our parameters do not load correctly or are not set.

```
$phoneDigits1 = $this->params->get('phoneDigits1', 4);
```

Now, assuming you set the digit parameters to 3 and 5 as per the previous screenshot, you can add a phone number in the format 123-45678 into one of your articles and you should see that they are now getting changed to a Click to Call link, as in the following screenshot:



So now your plugin is much more flexible as the number of digits is not hardcoded, and anyone using your plugin can make this minor change without having to touch any code.

If you wanted to make this plugin even better, you could make it recognize multiple phone number formats, because as you can see, if you set it to recognize the phone number format 123-45678, then it no longer changes the 1234-5678 numbers to Click to Call.

## Adding the language files

At the moment, all the text used in our plugin is hardcoded, so if a user wanted to change any of the text, they would have to edit the source code which is something we want to avoid for our end users. A lot of people use Joomla! in their native language and not everyone is going to want the text in English. Even if your plugin is only ever going to be used by English speakers, you still want them to be able to easily override the text to suit their needs. So we are going to create a language file, and instead of hardcoding the text, we will use a language string.

Firstly, we are going to replace the text in the parameters we have just added with language strings. Edit the `clicktocall.xml` file, and make the following changes:

```
<?xml version="1.0" encoding="UTF-8"?>
<extension
    version="3.0"
```

```

    type="plugin"
    group="content"
    method="upgrade">
<name>Content - Click To Call</name>
<author>Tim Plummer</author>
<creationDate>April 2013</creationDate>
<copyright>Copyright (C) 2013 Packt Publishing. All rights
reserved.</copyright>
<license> http://www.gnu.org/licenses/gpl-3.0.html</license>
<authorEmail>example@packtpub.com</authorEmail>
<authorUrl>http://packtpub.com</authorUrl>
<version>1.2.0</version>
<description>This plugin will replace phone numbers with click
to call links. Requires Joomla! 3.0 or greater.
Don't forget to publish this plugin!
</description>
<files>
    <filename plugin="clicktocall">clicktocall.php</filename>
    <filename plugin="clicktocall">index.html</filename>
</files>
<languages>
    <language tag="en-GB">language/en-GB/en-
GB.plg_content_clicktocall.ini</language>
</languages>
<config>
    <fields name="params">
        <fieldset name="basic">

            <field name="phoneDigits1" type="text"
                default="4"
                label="PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS1_LABEL"

description="PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS1_DESC"
            />
            <field name="phoneDigits2" type="text"
                default="4"
                label="PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS2_LABEL"

description="PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS2_DESC"
            />

        </fieldset>
    </fields>
</config>
</extension>

```

We have incremented the version number since we are adding a new feature, then included the language tags that tells Joomla! to load our language file from the /language/en-GB folder of our plugin install file and put them into the /administrator/language/en-GB folder when the extension is installed.

```
<languages>
  <language tag="en-GB">language/en-GB/en-
    GB.plg_content_clicktocall.ini</language>
</languages>
```

The format of the language string is PLG (because it is a plugin) followed by CONTENT (because this is a content plugin), followed by the plugin name CLICKTOCALL, then a short description of what this language string is for. Language strings are always in uppercase and can only contain alphabetic characters and underscores, you can't use spaces.



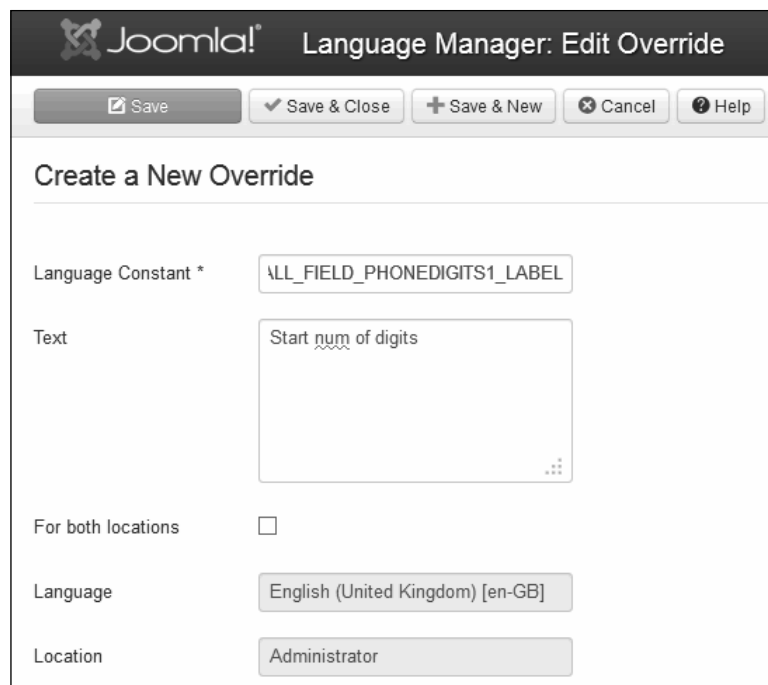
In Joomla! 1.5, you could use spaces and punctuation in language strings but Joomla! 1.6 changed to using the native PHP ini parser for language files which is much more strict but loads faster.

Create the file at /administrator/language/en-GB/en-GB.plg\_content\_clicktocall.ini which will contain the following language strings:

```
PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS1_LABEL="Digits 1"
PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS1_DESC="How many digits
in the first part of the phone number?"
PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS2_LABEL="Digits 2"
PLG_CONTENT_CLICKTOCALL_FIELD_PHONEDIGITS2_DESC="How many digits
in the second part of the phone number?"
```

Each line of the language file has the languages string followed by an equals sign, then the actual text enclosed in speech marks.

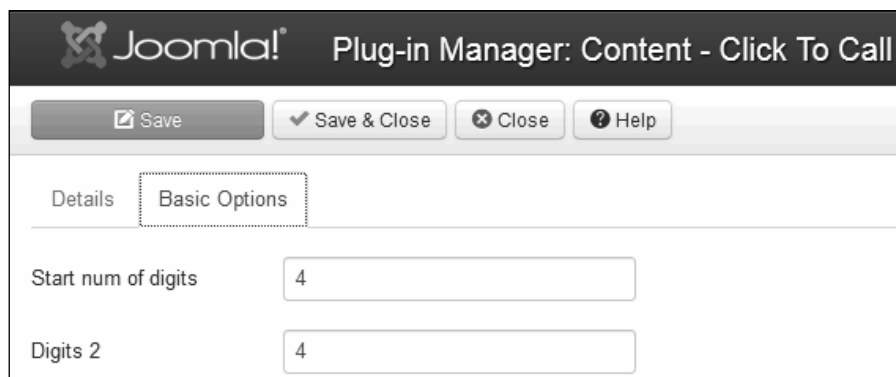
You can now try creating an Administrator language override. Make sure you set the filter to **English (United Kingdom)** and **Administrator** before creating the override so that it is for the backend. Override the language string PLG\_CONTENT\_CLICKTOCALL\_FIELD\_PHONEDIGITS1\_LABEL and change to Start num of digits.



The screenshot shows the Joomla! Language Manager: Edit Override interface. At the top, there's a Joomla! logo and the title "Language Manager: Edit Override". Below the title is a toolbar with buttons: Save, Save & Close, Save & New, Cancel, and Help. The main section is titled "Create a New Override". It contains the following fields:

- Language Constant \*: \LL\_FIELD\_PHONEDIGITS1\_LABEL
- Text: Start num of digits (in a text area)
- For both locations: ☐
- Language: English (United Kingdom) [en-GB]
- Location: Administrator

As you can see, the text **Digits 1** has now been replaced with the override **Start num of digits** without having to edit any source code. Alternatively, you could try changing the text by editing the file `/administrator/language/en-GB/en-GB.plg_content_clicktocall.ini` directly. Obviously you'd need to remove the language override, otherwise that will still take precedence.



The screenshot shows the Joomla! Plug-in Manager: Content - Click To Call interface. At the top, there's a Joomla! logo and the title "Plug-in Manager: Content - Click To Call". Below the title is a toolbar with buttons: Save, Save & Close, Close, and Help. The main section has two tabs: Details and Basic Options. The Basic Options tab is active, showing the following fields:

- Start num of digits: 4
- Digits 2: 4

Ideally all text displayed by your plugin should use a language string rather than being hardcoded. You will need to zip up your plugin again so the version you distribute will contain these new features. Before you zip it up, you will need to create the folder `language` in your plugin folder that you will be zipping, and within that, create an `en-GB` folder. Don't forget to include an `index.html` file in each folder. Now copy in the language file `/administrator/language/en-GB/en-GB.plg_content_clicktocall.ini` in the folder. The folder you are using to create the installable package should now have the following files:

1. `/language/en-GB/en-GB.plg_content_clicktocall.ini`
2. `/language/en-GB/index.html`
3. `/language/index.html`
4. `clicktocall.php`
5. `clicktocall.xml`
6. `index.html`

Zip it up and call this installable package as `plg_content_clicktocall_v1.2.0.zip`, and try it out on another Joomla! website. Even though we said this plugin was for Version 3.0 or greater, it will actually install and work perfectly on a Joomla! 2.5 site as all the code we have used is available in both versions, and the Joomla! version number in the installation XML file is not strictly enforced.

## Summary

Congratulations, you have now created your first extension for Joomla!: a plugin to change phone numbers in articles into Click to Call links. We learned about the installation XML file, and the PHP code required to make a simple plugin. We took a look at the different types of plugins in Joomla! and what events can trigger them. We then created an installable package of our plugin and tried it out on a Joomla! website. You can now add Joomla! Extension Developer to your resume!

In the next chapter, you will create your first Joomla! module.

## Where to buy this book

You can buy Learning Joomla! 3Extension Development from the Packt Publishing website: <http://www.packtpub.com/learning-joomla-3-extension-development/book>.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our [shipping policy](#).

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.



[www.PacktPub.com](http://www.PacktPub.com)

**For More Information:**

[www.packtpub.com/learning-joomla-3-extension-development/book](http://www.packtpub.com/learning-joomla-3-extension-development/book)