**DRUPAL ANSWERS**

**PATCH IN DRUPAL**

A patch is a file that contains a list of differences between one set of files, and another.  Through patches, all the changes in code like additions or deletions to Drupal core can be made. Also, patches can be used to make changes to another copy of the original or main file.

**Creating a patch**

For most improvements, use the following command after making your changes:

**git diff > [description]-[issue-number]-[comment-number].patch –use this command**

**Applying a patch**

Download the patch to your working directory. Apply the patch with the following command:

Git apply -v [patchname.patch]

To avoid accidentally including the patch file in future commits, remove it:

rm [patchname.patch]

Steps to download a patch:

Step-by-Step: Patch in a Contributed Module in drupal

Okay, now we are ready to patch a module. Here are the steps:

1. **Locate the desired patch**
   * Navigate to the appropriate issue on drupal.org
   * Make sure you are choosing a patch that is designed for your version of the module
2. **Right-click the patch and save it to the module's directory for your Drupal install**
   * When you right click, you will see an option to 'save link as...' (in Firefox or chrome in windows). Choose this and save the file to the right spot.
3. **Patch the file.** Now you are ready to run the patch command and patch the file.
   * You can execute a test of installing the patch by entering:   
     patch -p0 --dry-run < fix\_scary\_module.patch   
     This prints the results of applying the patch without actually making any changes.
   * When ready to do the patch for real, enter **one** of the following at the Cygwin prompt:   
     patch -p0 < fix\_scary\_module.patch OR   
     patch -p0 -b < fix\_scary\_module.patch   
     (Don't enter both commands - just one.) The version with the '-b' option will automatically create backups of any files that the patch modifies.  
     Note: you do not need to run update.php after applying a patch.
4. **See if the patch applied.**
   * If the patch appears to have worked, you should now visit your site and reload some pages.
   * First make sure that the site appears to be working normally in all respects.
   * Then check to see if the desired new feature or fix has been enabled and that it does what it was supposed to do.
   * In some cases, you may need to clear cache to see changes.
   * If it works in all respects, then you are good to go and are done.
5. **Reverting back to the original file.**
   * If your patch successfully applied and works, please ignore this step. If your patch either failed to apply (step #4) or appeared to apply, but didn't actually change site behavior (step #5), then you should revert to the original .module version as follows:
   * Go back to your cygwin window.
   * At the module's directory enter:  
     patch -p0 -R < fix\_scary\_module.patch.

**2) Drupal page request process:**

**We have to right the code:**

1. Define ('DRUPAL\_ROOT', getcwd ());  
   require once DRUPAL\_ROOT . '/includes/bootstrap.inc';  
   drupal\_bootstrap(DRUPAL\_BOOTSTRAP\_FULL);  
   menu\_execute\_active\_handler();

Yes, lines of code returns every single page on your Drupal website.

A call is made to getcwd () and output is assigned to the 'DRUPAL\_ROOT' constant, basically with this Drupal is telling itself where it is installed.

**3)Creating custom module:**

Create Custom Module

In modules directory *(sites/all/modules)* create a folder for your module.

For this example our module will be called*myblockmodule*, so we would create the directory*sites/all/modules/myblockmodule*.

1. **Create a .info file**

All modules in Drupal require a .info file to let Drupal know that our module exists, what it's name is, as well as other information about our module. In your module folder (sites/all/modules/myblockmodule), create a file named myblockmodule.info. In myblockmodule.info enter the following:

name = MyBlockModule

description = This module provides a custom block with dynamic content.

package = MyModules

core = 7.x

2. Create a .module file

There are two files which every module must have. First is the .info file, which we already created. The second is the .module file. Despite the file's extension it is really a PHP script. This file is evaluated every time Drupal a Drupal page is load. The main purpose of this file is to define functions which implement hooks. Hooks are evaluated during different Drupal events and can be called by Drupal itself or other custom modules.

<?php

/\*\*

\* Declare what blocks are provided by this module.

\* Implements hook\_block\_info().

\*/

function myblockmodule\_block\_info(){

$block['marty\_countdown'] = array(

'info' => t('Marty Countdown'),

'cache' => DRUPAL\_NO\_CACHE,

);

return $block;

}

/\*\*

\* Define what our block is going to look like.

\* Implements hook\_block\_view().

\*/

function myblockmodule\_block\_view($block\_key){

$block = array();

if($block\_key == 'marty\_countdown'){ //We only want to define the content of OUR block

//This is the title of the block.

$block['subject'] = t('Marty McFly Countdown');

//Define the block content.

$block['content'] = t('Marty will arrive in ').(strtotime("2015-10-21")-time()).t(' seconds.');

}

return $block;

}

4) **Drupal seo module:**

[Page Title](https://www.drupal.org/project/page_title)

Page Title allows you to take control of the node, and creates whatever title name you want. By setting patterns for the title, you can create unique and relevant title for the content. You can also set different patterns if you choose page title as automatically being generated.

[Metatag](https://www.drupal.org/project/metatag)

Drupal doesn’t support editable meta tags fields. This module enables your control over the meta tags, including page title, description, and keywords. You are able to set meta tags for users, nodes, taxonomy terms, and views,…  
Meta tags also gives users options to set default meta tags for the entire site - whole or individual pages.

[Search 404](https://www.drupal.org/project/search404)

The search 404 module gives you a way to handle with 404 page errors on your website. Upon installing the modules, when a user enters a URL that renders a 404 page, the result be will automatically navigated to the internal site search with the related term in the URL.

[**Redirect**](https://www.drupal.org/project/redirect)

This is a great module if you want to deal with duplicate content.

[**Global Redirect**](https://www.drupal.org/project/globalredirect)

The module also comes with other important features: removing the trailing slash in the URL; verifying that clean URLs are being implemented correctly; checking permission and access to nodes, URLs.

[**Content Optimizer**](https://www.drupal.org/project/contentoptimizer)

Content Optimizer brings a quick analysis and statistics of your website content and gives you a specific idea about how SEO friendly your site is. And it provides recommended actions to improve search engine rankings. The Content Analysis module is required for installation.

[**SEO Checklist**](https://www.drupal.org/project/seo_checklist)

The SEO checklist modules lists the most important SEO tasks and the relevant modules needed installing to improve onsite SEO. The module doesn’t take any action with your site. It only informs a comprehensive SEO tasks you need to do.

[**Drupal SEO Tools**](https://www.drupal.org/project/seotools)

This is an “all in one” SEO suite. Drupal SEO Tools provides a dashboard which covers a number of SEO functions for the sites including: keywords, titles, tags, paths, redirects, sitemaps, Google analytics, webmaster tool, etc.

[**Menu Attributes**](https://www.drupal.org/project/menu_attributes)

The Menu attributes module allows the administrator to pinpoint specific attributes, consisting id, name, class, styles, and rel.

[**Links Checker**](https://www.drupal.org/project/linkchecker)

Broken links are precisely a negative signal to search engines. You’ll need a module like Links Checker to occasionally specify failed results which you need to make a fix to.

[**Site Verification**](https://www.drupal.org/project/site_verify)

In order to make sure that search engines are properly crawling and indexing your site, you have to verify it. This module makes the verifying process easier and it supports a number of search engines – Google, Yahoo, Bing,… The module enables verifying by uploading a html file or adding meta tags.

[**Html Purifier**](https://www.drupal.org/project/htmlpurifier)

Search engine reads your website content via raw html. So it’s always important to make sure have a clean easy-to-read code. Html filter will check your website code to conform to standard html rule. It also helps removes malicious code along the way.

[**Taxonomy Title**](https://www.drupal.org/project/taxonomy_title)

With Taxonomy Title, you can edit the heading tag (H1) of the taxonomy page. The importance of H1 tag should never be neglected for on page SEO. And we have a whole module dedicated to change the H1 tag of the taxonomy.

5) **Druapl ecm, cms and wcm differences**

A Web content management system (WCM, WCMS or Web CMS) is content management system (CMS) software, implemented as a Web application, for creating and managing HTML content. It is used to manage and control a large, dynamic collection of Web material (HTML documents and their associated images). A**WCMS** facilitates content creation, content control, editing, and essential Web maintenance functions.

A presentation layer displays the content to Web-site visitors based on a set of templates. The templates are sometimes XSLT files.  
  
Most systems use server side caching boosting performance. This works best when the WCMS is not changed often but visits happen on a regular basis.

**CMS:**

A content management system (CMS) is a collection of procedures used to manage work flow in a collaborative environment. These procedures can be manual or computer-based. The procedures are designed to:

* Allow for a large number of people to contribute to and share stored data
* Control access to data, based on user roles. User roles define what information each user can view or edit
* Aid in easy storage and retrieval of data
* Reduce repetitive duplicate input
* Improve the ease of report writing
* Improve communication between users.

**ECM:**

Enterprise Content Management" or ECM minimally encompasses Document Management, Collaboration, Records Management and Web Content Management.

**6)Drupal creating web service and testing service :**

**Testing with REST Server**

To get the most out of this page, you should first take a look at [Working with REST Server](http://drupal.org/node/783254)to get more of a conceptual overview on how the REST Server works.

As described on that page, there are six types of resources you can define

* [Retrieve](https://www.drupal.org/node/790416#retrieve)
* [Create](https://www.drupal.org/node/790416#create)
* [Update](https://www.drupal.org/node/790416#update)
* [Delete](https://www.drupal.org/node/790416#delete)
* [Action](https://www.drupal.org/node/790416#action)
* [Targeted Action](https://www.drupal.org/node/790416#targeted_action)

We will test each one with the user service. When testing, use the URL you figured out in the previous section.

Retrieve

Retrieving information from your resource is very simple. The format is

http://<url>/<identifier>.<format>

Identifier is the unique identifier of the data you are attempting to retrieve (in this example, the user ID.) Format is one of the formats the REST server supports returning data in.

Lets use XML for the time being. So to test the user resource, enter the following

http://<url>/1.xml

**Create**

In order to create through a resource, you have to POST data to the URL. For posting, you just use the base URL without the identifier and format.

The steps below are for Poster plugin for Firefox, but they are similar to Advanced REST Client for Chrome.

1. After installing Poster you will need to reboot Firefox. Once you have rebooted you can active Poster from the menu at Tools->Poster.
2. In the URL field, enter the base URL for your resource.
3. Click the Parameters tab. This is where you enter the data to be submitted, which mimics the fields that would be posted from a web form.
   * name (username)
   * pass (plain text password)
   * mail (email address)

Enter a key/value pair and click the Add/Change button to add it to the list**.**

Now click back to the Content to Send tab and click Parameter Body. This will convert your Parameters to properly encoded data, and change the Content Type to 'application/x-www-form-urlencoded'.

Under Actions you'll see several buttons. Create requests are POSTed to the resource, so click Post. Ideally, if all went well, you will see something like the following:

This is an XML representation of the user you just created.

If you would rather test from code, you can do this using a Drupal bootstrap script with the following code. The $account array created here is just an array representation of the parameters above.

$account = array(

'name' => 'test\_user',

'pass' => 'xyzzy',

'mail' => 'test\_user@xyzzy.com',

);

$response = drupal\_http\_request($url, $headers, 'POST', http\_build\_query($account, '', '&'));

If your test did not succeed, here are some things to check

* Are your parameters named correctly? They have to be exactly right or the submission will not validate.
* Load your server's home page. Often a failed form submission will trigger a drupal\_set\_message(), however these messages are only mirrored to the screen, not to our service's response. You will often not see these messages until your next page load.
* Check the dblog on your server to see if any errors occurred.
* For users specifically, note that two users can not have the same username or email. If you resubmit with data that you previously submitted, your submission will fail.

These will often provide clues to your problem.

**Update**

Update functions work exactly the same as Create functions, except for a couple of small differences.

* The URL you submit to has the unique identifier appended to it
* Instead of using POST you will use PUT.

So in the above example, you can see in the response dialog that the UID of our new user is 7. So we can add this into our URL, and perhaps we want to change the user's email address as demonstrated below.

**Delete**

Delete functions work exactly the same as Update functions except for a couple of small differences

* You don't actually need to submit any parameters (you can submit them but they will be ignored.)
* You use DELETE instead of PUT.

**6)DRUPAL VERSIONS:**

DRUPAL5-

DRUPAL6-

DRUPAL7-

DRUPAL8-

It is consists of hooks A hook is a PHP function that is named foo\_bar(), where "foo" is the name of the module (whose filename is thus foo.module) and "bar" is the name of the hook. Each hook has a defined set of parameters and a specified result type.

**DRUPAL 7 FUCTIONS**

|  |
| --- |
| [custom\_url\_rewrite\_inbound](https://api.drupal.org/api/drupal/developer%21hooks%21core.php/function/custom_url_rewrite_inbound/6.x) |
| [custom\_url\_rewrite\_outbound](https://api.drupal.org/api/drupal/developer%21hooks%21core.php/function/custom_url_rewrite_outbound/6.x) |
| [hook\_access](https://api.drupal.org/api/drupal/developer%21hooks%21node.php/function/hook_access/6.x) |
| [hook\_actions\_delete](https://api.drupal.org/api/drupal/developer%21hooks%21core.php/function/hook_actions_delete/6.x) |
| [hook\_action\_info](https://api.drupal.org/api/drupal/developer%21hooks%21core.php/function/hook_action_info/6.x) |
| [hook\_action\_info\_alter](https://api.drupal.org/api/drupal/developer%21hooks%21core.php/function/hook_action_info_alter/6.x) |
| [hook\_block](https://api.drupal.org/api/drupal/developer%21hooks%21core.php/function/hook_block/6.x) |
| [hook\_boot](https://api.drupal.org/api/drupal/developer%21hooks%21core.php/function/hook_boot/6.x) |