Simple Magento Staging / Development Server Set-up

Foreword

This tutorial assumes you have a basic working knowledge of Magento and uses 2 domain names for demonstration purposes. Example.com refers to your live domain and dev.example.com refers to your development domain. So if your live domain is called widgets.com then the dev domain would be called dev.widgets.com.

This tutorial can be extended to many different scenarios such as handling multiple base currencies but most importantly it is a guide to setting up a staging (development) / live environment that allows you to switch between versions of Magento without impacting a 'live', production site.

This is great for seamlessly upgrading Magento on the fly, or testing a new Beta version of Magento without affecting your live site.

Set up Magento

Note: this step isn't necessary in configuring a development and live environment but demonstrates the flexibility of Magento and the techniques contained within this article.

Firstly, login to Magento and navigate to **system->manage stores**. Next, assign stores and store views to the websites for dev.example.com and example.com. You should end up with something like:

Website Name	Store Name	Store View Name
Live	Live Store	Live View 001
Dev	Dev Store	Dev View 001

Next go to: **system->configuration->web** and use the configuration scope drop down to select the 'Live' website. This will allow you to set a base URL for the website. So go ahead and set the base URL for Live site to http://example.com/ click save and then do the same for the 'Dev' website but use http://dev.example.com.

By doing this, Magento will now have 2 websites to work with which will be useful for testing out new themes etc before deploying to the live site.

Setting up the server

Symlinks

The most important part of the Magento multi website server set up is using Symlinks (symbolic references) to hook into the Magento core folders and files.

The following server directory structure uses the multi-currency example.com website to illustrate the idea:

/server-root/public_html/magento-live – This folder contains the 'live' version of Magento core.
/server-root/public_html/example.com/ - The 'live' domain Example.com is pointed at this folder.
/server-root/public_html/dev.example.com/ - dev.example.com/ is the root folder for your development site.

Firstly, you need to create symlinks within the example.com/ folder and the dev.example.com/ folder to point to the Magento core folders.

I use SSH for this using the MAC OSX terminal. Here are the commands I would use to link the folders that need to be linked. an example symlink:

```
cd /server-root/public_html/example.com/

ln -s /server-root/public_html/magento-live/app ./app
ln -s /server-root/public_html/magento-live/includes ./includes
ln -s /server-root/public_html/magento-live/js ./js
ln -s /server-root/public_html/magento-live/lib ./lib
ln -s /server-root/public_html/magento-live/media ./media
ln -s /server-root/public_html/magento-live/report ./report
ln -s /server-root/public_html/magento-live/skin ./skin
ln -s /server-root/public_html/magento-live/var ./var
```

This will create linked folders that are similar to 'shortcuts' on a desktop PC that reference the Magento core folders contained in the magento-live folder.

You will need to create symlinks for each folder / website you use with Magento i.e. you will now need to create symlinks within the dev.example.com folder.

After completing these steps you will have 2 folders that point to the magento-live core installation.

Now you have created symlinks to Magento core you will need to make copies of the Magento core .htaccess and index.php files and place them in the example.com and dev.example.com/ folders. This will instruct Magento to treat your newly created sub-directories as stand alone websites.

.htaccess

Open up your FTP client and navigate to the Magento core folder (/server-root/public_html/magento-live) and download both the .htaccess and index.php files.

Server Rewrites

Next, edit the .htaccess file by making the following amends:

Uncomment and change the line that reads: #RewriteBase / to RewriteBase / - delete the hash. This will mean that all requests to / will be treated as the base URL. This edit applies to both example.com and dev.example.com.

Site-maps

If you are using site-maps you can provide a simple way to include them from your core Magento installation like so:

```
RewriteRule  ^sitemap.xml/?$ /pathtoyourmagentofolder/sitemaps/whichsite/sitemap.xml
```

This means that all requests to example.com/sitemap.xml will be re-written to the /pathtoyourmagentofolder/sitemaps/whichsite/sitemap.xml path.

Fooman Speedster

If you are using Fooman speedster to minify your CSS and JS files (highly recommended) then you can include something like the following to include the Fooman minification scripts:

```
############################################ ## Compress, Combine and Cache Javascript/CSS
RewriteRule ^(index.php/)?minify/([^/]+)(/.*.(js|css))$ pathtoyourmagentofolder/lib/minify/m.php?f=$3&d=$2
```

Done editing .htaccess

You are now done editing the .htaccess file so upload it to the /server-root/public_html/example.com/ and /server-root/public_html/dev.example.com/ folders.

Index.PHP

Now you have created the symlinks and edited the ./htaccess file for each website / folder you can edit the index.php file. This is the Magento bootstrap – the file that initializes Magento.

You will need an index.php file for each domain and 2 simple edits need to be completed to include Magento core and your new currency specific website.

Find the lines:

```
$mageRunCode = isset($_SERVER['MAGE_RUN_CODE']) ? $_SERVER['MAGE_RUN_CODE'] : '';
$mageRunType = isset($_SERVER['MAGE_RUN_TYPE']) ? $_SERVER['MAGE_RUN_TYPE'] : 'store';
```

And edit these to something like the following:

```
$mageRunCode = isset($_SERVER['MAGE_RUN_CODE']) ? $_SERVER['MAGE_RUN_CODE'] : 'websitecode';
$mageRunType = isset($_SERVER['MAGE_RUN_TYPE']) ? $_SERVER['MAGE_RUN_TYPE'] : 'website';
```

In the above example, websitecode would be the code used to define your website as set within the Magento administration interface under system->manage stores.

Conclusion

Using this simple approach you can bring great flexibility to serving different version of Magento depending on your requirements. A typical purpose for this technique is when upgrading Magento to a new version.

This can be accomplished using the following as a rough guide:

1. Back up the existing database (db1), themes and modules.
2. Create an upgrade folder i.e. magento1324/.
3. Create a new database (db2) using the data from the database backup (db1).
4. Using the symlink technique above, navigate to your dev.example.com/ folder and edit symlinks to point at your new magento1324/ folder. This will allow you to preview a new version of Magento without affecting your existing live site. If you want to upgrade the existing site you would simply follow these steps but edit the symlinks for example.com (your live site) instead of dev.example.com (your dev site).
5. Download and install a newer version of Magento i.e. 1.3.2.4 into the magento1324/ folder using the new database (db2).
6. Check everything is working.