```
# Disable Directory Listing
Options -Indexes

# The first line sets the environment up to follow symbolic links using the
# Options directive. This may or may not be necessary, but some web hosts use
# symlinks (similar to alias in MacOSX or shortcuts is Windows) for common
# HTTP request errors and these are usually symlinked files, or at least this
# is how I understand the reasoning.
Options +FollowSymLinks

# Use the Rewrite Engine
RewriteEngine On

# The next two lines are very, very important it restricts rewriting URLs only
# to paths that do not actually exists. This prevents the rules below from
# matching example.com/images/logo.png for example. The first prevents
# existing directories with the !-d flag and the second with !-f means ignore # existing
files.
RewriteCond %{SCRIPT_FILENAME} !-d
RewriteCond %{SCRIPT_FILENAME} !-f

# The next three lines are the actual URL rewriting commands. Each line
# creates a rule that tries to match a regular expressions pattern against the
# incoming URL. Regular expressions, at least for me, are a hard set of rules
# to remember but I always find it helpful to use this tutorial
# <http://blog.themeforest.net/screencasts/regular-expressions-for-dummies/>
# by Nettut's own Jeffery Way and the tool <http://gskinner.com/RegExr/> he
# recommends. I found it easy to type in sample URLs we want to match and the
# try to hack together the pattern.
#
# The first argument is the pattern, between the caret and dollar sign. We
# tell Apache we want URLs asking for the users directory (an artificial
# directory, doesn't have to actually exist) followed by a / and any length of
# numbers. The parenthesis create a capture group, you can use as many of
# these as you want, they serve as variables that we can then transplant into
# our rewrite. The asterisk means the user can enter whatever they want, and
# it won't affect the rewrite, this is primarily to handle a trailing slash so
# example.com/users/123 is the same as example.com/users/123/ as users would
# expect.
#
# The second argument is the path we want to actually call, this unlike the
# The second argument is the path we want to actually call, this unlike the
# first must be a real file. We tell Apache to look in the current directory
# for a file called profile.php and send the parameter id=$1 along with it.
# Remember the capture group earlier? That is where we get the variable $1,
# capture groups start at one. This creates a URL on the server like
# example.com/profile.php?id=123.
RewriteRule ^(.*/?$) public-index.php?p=$1

# For Products
#RewriteRule ^product/(.*/?$) ./system/views/product.php?p=$1
#RewriteRule ^product/(.*/?$) ./shadow/system/views/product.php?p=$1

<IfModule mod_rewrite.c>
# For sales:
RewriteRule ^shop/sales/?$ sales.php
# For the primary categories:
RewriteRule ^shop/([A-Za-z\+]+)/?$ shop.php?type=$1
# For specific products:
RewriteRule ^browse/([A-Za-z\+\-]+)/([A-Za-z\+\-]+)/([0-9]+)$ browse.php?
type=$1&category=$2&id=$3
</IfModule>

# force url to lowercase if upper case is found
RewriteCond %{REQUEST_URI} [A-Z]
# ensure it is not a file on the drive first
RewriteCond %{REQUEST_FILENAME} !-s
RewriteRule (.*) rewrite-strtolower.php?rewrite-strtolower-url=$1 [QSA,L]
```