# Magento for Designers: Part 6

Magento is a stunningly powerful e-commerce platform. In this miniseries, we’ll learn how to get started with the platform, get to know the terminologies, set up a store and all related aspects of it and finally learn how to customize it to make it our very own.

In this sixth part, we’ll be building one of the most reader requested feature: getting a featured product slider in the home page. Excited? Let’s get started!

## The Full Series

* [Part 1: Installation and Setup](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-i/)
* [Part 2: Products, Taxes, Categories, Payment Gateways, etc.](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-2/)
* [Part 3: Theming](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-3/)
* [Part 4: Building the Theme](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-4/)
* [Part 5: Building the Product Information Page](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-5/)
* [Part 6: Building the Home Page, with a Content Slider](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

## A Quick Recap

In the last part, we built the product information page and took a good look at how data is extracted and displayed in a typical Magento theme.

We also took a look at the various core methods that our theme uses to get various media and data about our product. I hope you remember at least a sliver of that information because we’ll be using it again today.

## What are We Building Today?

Today, we’ll build the all important home page. As always, I’ll keep it as simple as possible and omit any extraneous features. Different stores have different content in their home pages. One of the most popular and definitely one of the most requested features is displaying a JavaScript powered content slider featuring special featured products. I know it sounds interesting and you’re itching to get started. Let’s get started!

## Goals for the Page

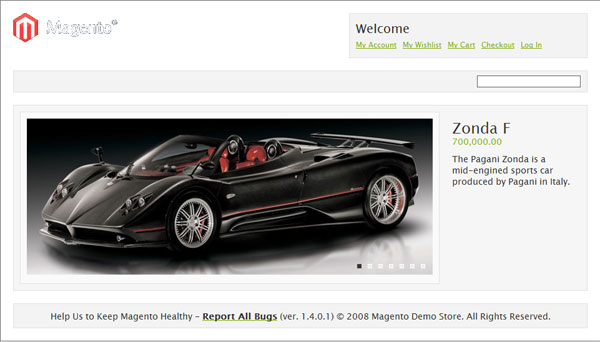
As with our other pages, our goals for this specific page are relatively simple. We want a simple slider that displays specific products in a neat slider. Thus, I’ve drawn up a short list of elements that I think it needs:

* JavaScript slider on the home page
* Slider needs a catchy image on the left to attract attention to the product
* In the right portion, we’ll be including the product’s name, price and a short description
* A mini navigation within the slider to switch between different slides
* And most important of all, not rely on an extension to obtain the featured items

That’s it. I want to keep it as simple as possible and thus opted to not over do it. We will of course need to work with both the front end and back end to accomplish all our goals.

## The Basic Look

The page basically has to look like so:



## Step 1 – The HTML

We’ll first look at the HTML for the content part alone. I’m assuming you’re fairly fluent in HTML and CSS so I’ll skip to the fairly important parts.

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. <div id="home-slideshow">
3. <div class="slide">
5. <div class="product-image">
6. <a href="#" title="Zonda F">
7. <img src="images/zonda.jpg" alt="Zonda F" title="Zonda F" />
8. </a>
9. </div>
11. <div class="product-details">
12. <h2>Zonda F</h2>
13. <div class="price">700,000.00 </div>
14. <p>The Pagani Zonda is a mid-engined sports car produced by Pagani in Italy. </p>
15. </div>
17. </div>
19. <!-- More slides for successive products -->
21. </div>

At this point, this is all pretty basic. home-slideshow is the container we’re wrapping the entire slideshow with and will be passed on to the JavaScript responsible for the slider.

Each slide is wrapped by a div called slide. Inside we have two floated containers: one for the image and the second for the information about the product itself.

That’s about all there is to it. We can move on to the styling part now.

## Step 2 – The CSS

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

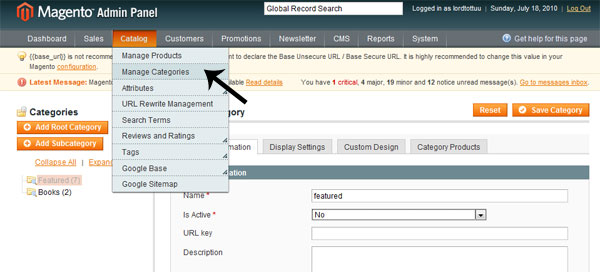
1. /\* Home page \*/
3. .slide {
4. clear: both;
5. overflow: auto;
6. padding: 10px;
7. width: 898px;
8. border: 1px solid #E1E1E1;
9. background: #F6F6F6;
10. }
12. .product-image {
13. float: left;
14. width: 650px;
15. padding: 10px;
16. border: 1px solid #E1E1E1;
17. background: #FAFAFA;
19. }
21. .product-details {
22. float: left;
23. width: 190px;
24. margin: 0 0 0 20px;
25. }
27. .product-details .price {
28. margin: -15px 0 0 0;
29. color : #7db000;
30. }
32. .product-details p{
33. margin: 10px 0 0 0;
34. }
36. #nav {
37. position: absolute;
38. top: 255px;
39. rightright: 250px;
40. z-index: 2000;
41. }
43. #nav a {
44. margin-right: 10px;
45. height: 5px;
46. width: 5px;
47. z-index: 1000;
48. float: left;
49. text-indent: -9999px;
50. cursor: pointer;
51. border: 1px solid #fff;
52. background:  #eee;
53. }
55. #nav a:hover {
56. border: 1px solid #fff;
57. background:  #999;
58. }
60. #nav a.activeSlide {
61. border: 1px solid #444;
62. background:  #333;
63. }

Nothing fancy here. Very basic CSS to place the elements in position.

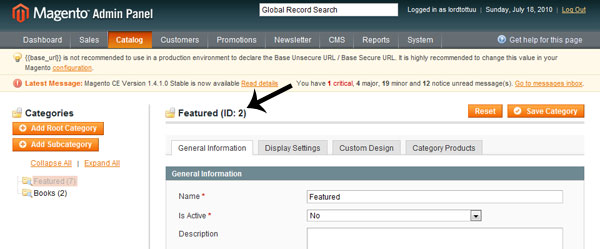
The only point to note is the declaration for the class activeSlide. This class is assigned to the current slide’s respective navigation anchor.

## Step 3 – Setting up Our Store

The first thing we need to do is setup a separate category to which we can assign all the products that we need to appear on our slider. Click on the catalog menu and then on categories. Choose add new root category.



Key in a name of your choice and then choose no for the is active dropdown. We don’t want it displaying in other parts of the site.



Once the new category has been saved, you can find the ID it has been assigned to. Note this down. We’ll need it for later.

## Step 4 – Creating Our Core Code

We’ll be using a single template file to pull out the relevant information, format it as needed and display everything. You have complete freedom regarding the name and location of this file but in the interest of keeping it simple, I’ve named it featured.phtml and placed it in catalog/product/featured.phtml. Remember, this goes into your template folder, not your layout folder.

The contents of the file looks like so. Don’t worry about the length, we’ll look at each part works after the code.

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. <?php
2. $cat\_id = "2";
3. $\_productCollection = Mage::getResourceModel('catalog/product\_collection')
4. ->addAttributeToSelect(array('name', 'price', 'small\_image', 'short\_description'), 'inner')
5. ->addCategoryFilter(Mage::getModel('catalog/category')->load($cat\_id));
7. if(!$\_productCollection->count()):
8. echo $this->\_\_('There are no products matching the selection.');
9. else: ?>
11. <div id="home-slideshow">
12. <?php $i=0; foreach ($\_productCollection as $\_product): ?>
14. <div class="slide">
16. <div class="product-image">
17. <a href="<?php echo $\_product->getProductUrl() ?>" title="<?php echo $this->htmlEscape($this->getImageLabel($\_product, 'small\_image')) ?>">
18. <img src="<?php echo $this->helper('catalog/image')->init($\_product, 'small\_image')->resize(650, 250); ?>" alt="<?php echo $this->htmlEscape($this->getImageLabel($\_product, 'small\_image')) ?>" title="<?php echo $this->htmlEscape($this->getImageLabel($\_product, 'small\_image')) ?>" /></a>
19. </div>
21. <div class="product-details">
22. <h2><?php echo $\_product->getName(); ?></h2>
23. <div class="price"><?php echo number\_format($\_product->getFinalPrice(),2);?> </div>
24. <p><?php echo $\_product->getShortDescription(); ?></p>
25. </div>
27. </div>
28. <?php endforeach ?>
30. </div>
32. <?php endif; ?>

Here’s the first piece:

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. $cat\_id = "2";

First, we’ll need to specify the ID of the category from which we want to pull in our products. In our case, this will be 2, the ID of the category we just created.

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. $\_productCollection = Mage::getResourceModel('catalog/product\_collection')
2. ->addAttributeToSelect(array('name', 'price', 'small\_image', 'short\_description'), 'inner')
3. ->addCategoryFilter(Mage::getModel('catalog/category')->load($cat\_id));

This looks slightly complicated but let me assure you, it’s very simple. To put it in layman’s terms, we essentially pass in the ID of the category we need through the variable cat\_id and ask Magento to retrieve the name, price, image and short description of the products that match. Right now, I’d advise you to not fiddle with the rest unless you’re well versed in MVC and other aspects of Magento’s architecture.

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. if(!$\_productCollection->count()):
2. echo $this->\_\_('Sorry, no items matched your selection criteria');
3. else: ?>

We will need to check whether we have matching items, don’t we? That’s what we’re doing here. Just making sure we have matching items. If not, we display an error.

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. <?php $i=0; foreach ($\_productCollection as $\_product): ?>

Let’s start looping through our product collection!

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. <a href="<?php echo $\_product->getProductUrl() ?>" title="<?php echo $this->htmlEscape($this->getImageLabel($\_product, 'small\_image')) ?>">
2. <img src="<?php echo $this->helper('catalog/image')->init($\_product, 'small\_image')->resize(650, 250); ?>" alt="<?php echo $this->htmlEscape($this->getImageLabel($\_product, 'small\_image')) ?>" title="<?php echo $this->htmlEscape($this->getImageLabel($\_product, 'small\_image')) ?>" /></a>

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. <h2><?php echo $\_product->getName(); ?></h2>
2. <div class="price"><?php echo number\_format($\_product->getFinalPrice(),2);?> </div>
3. <p><?php echo $\_product->getShortDescription(); ?></p>

If you’ve read the previous part of this series, this code must be pretty familiar to you. We use Magento’s baked in methods to retrieve media pertaining to our product.

The second snippet may look a little different because instead of using slightly more abstracted methods, I used closer-to-bone methods to directly acquire the data we need.

If you’re feeling a little confused here, just go through the above code one more. The name of the methods are pretty explanatory. The only stumbling block should be how we acquire the image itself: we ask one of our helpers to acquire the image and resize it to the size we need. Nothing more to it.

Note that we open and close a slide div each time our loop runs. This is to encapsulate each product’s information with a container element to make it easy for us to create a slider later.

And with this, our base PHP code is complete.

## Step 5 – The JavaScript

It makes no sense to completely reinvent the wheel here since the core focus of this series is on Magento. Thus, we’ll be using an excellent plugin to create our slider today: [the jQuery Cycle plugin](http://jquery.malsup.com/cycle/).

But first, go to root/js and create a folder named cirrus. Different people have different methods of organizing here. I prefer having a theme’s JS separate instead of organizing them by source. We’ll need a copy of jQuery [1.4 hopefully], the latest cycle plugin and a custom script file, which I’m naming mocha.

Here’s the content of our JS file.

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. (function($) {
3. $(document).ready(function(){
4. $('#home-slideshow')
5. .before('<div id="nav">')
6. .cycle({
7. fx: 'fade',
8. speed:    1500,
9. timeout:  5000,
10. pause: 1,
11. delay:  5000,
12. pager: '#nav'
13. });
14. });
15. })(jQuery)

First up, notice that we’re wrapping everything inside a (function($) { //Code })(jQuery) block. This is make sure jQuery doesn’t conflict with any other included library.

The document ready block, as always, is to make sure that our code runs only after all the page’s assets are loaded since Magento likes to put our JavaScript files at the top.

Initializing the plugin is pretty simple. Just type in the ID of the main container, home-slideshow in our case, and away we go. I’ve just tweaked some of the options to make it do our bidding.

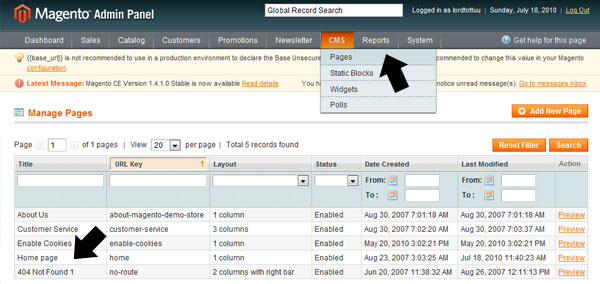
Something to take note of is that we’re dynamically creating and positioning our container for the navigation elements inside the DOM with JavaScriptS. This element is only needed if JavaScript is enabled. If it isn’t, we don’t want empty, useless code in our page.

We also pass in the selector the navigation container to the slider plugin.

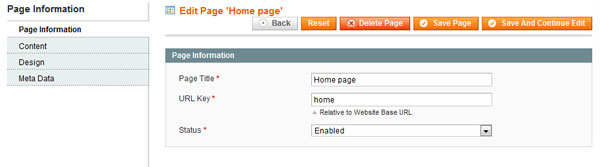
That’s it. The JavaScript part of our code is done.

## Step 6 – Setting up Our Home Page

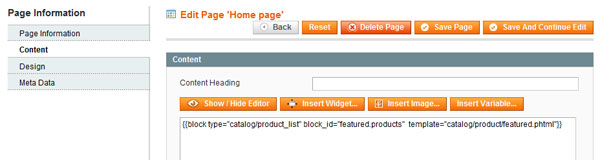
Now that all our groundwork has been completed, we can finally configure our home page now.



Click on the CMS menu and choose the pages option. Click on home page in the resulting page.



The first tab should be pretty self explanatory. Make sure that our page is enabled.

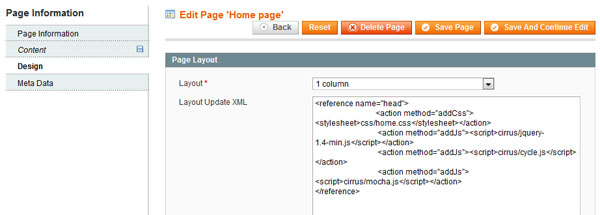


In the content tab, make sure you enter the following after disabling the editor:

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. {{block type="catalog/product\_list" block\_id="featured.products"  template="catalog/product/featured.phtml"}}

This is Magento’s method of including required blocks. We just give it an ID to refer it by and point it to our custom template.



In the design tab, change the layout template to 1 column since we’re using it for everything and then enter the following into the layout update xml field.

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-6/)

1. <reference name="head">
2. <action method="addCss"><stylesheet>css/home.css</stylesheet></action>
3. <action method="addJs"><script>cirrus/jquery-1.4-min.js</script></action>
4. <action method="addJs"><script>cirrus/cycle.js</script></action>
5. <action method="addJs"><script>cirrus/mocha.js</script></action>
6. </reference>

Remember how when building the product page, we included custom CSS in the page’s layout file to make sure the entire codebase isn’t loaded for every page? We’re doing essentially the same here. Instead of creating a XML file just for this, Magento lets us enter this directly through the interface.

We just load our custom CSS, our JS libraries and the script file that contains all our code.

Click on “save” to save up all our progress, empty Magento’s cache and watch the beautiful slider on your home page slide through in its excruciatingly lovely glory!

## What We’ll be Building in the Next Part

.. is completely up to you. I’ve already covered almost all relevant information for customizing any view and going through each and every view is completely counter-intuitive. Thus, I’ve concluded that all that’s left is how to create a custom module and I think by then this series will have runs its course. I may write a quick Magento tweaks and optimization how-to if enough people show interest for such an article.

If you do feel that something else needs to be covered before this series comes to a close, don’t hesitate to leave a comment below to let me know!

## The Last Word

And we are done! Today, we learnt how to create a custom home page complete with a slider displaying featured products. We also learnt to do all this without relying on a third party extension and instead coding all this ourselves. Hopefully this has been useful to you and you found it interesting. Since this is a rather new topic for a lot of readers I’ll be closely watching the comments section so chime in there if you’re having any doubts.

Questions? Nice things to say? Criticisms? Hit the comments section and leave me a note. Happy coding!