Quic Start to Magento Customization

Part 1: Request Flow and Controller

A lot of unexplained code and convention. We’ll take a slightly different approach. We’ll start by creating a simple module and we’ll build on it. We’ll concentrate on

**How to add your module to the system**

**What configuration files you need and why**

**What a controller is in magento and how you can work with it**

We’ll learn how magento detects modules and processes requests (request flow)

We’ll also go round Routes, routers, and what makes your controller run

The main building block in magento is module. In magento module is a complex concept, even a core engine is a black box. And it takes its place among the other modules. This is different from many other architectures for example simfoni. That’s why to add functionality we typically create a module and here we refer to modules. To add functionality in magento is also useful to understand magento’s concepts of code pools. We put all magento modules in one place the app/ folders also pronounced

As ‘app/code’. If you take a close inside a code folder you’ll see three folders:

Community, core, local.

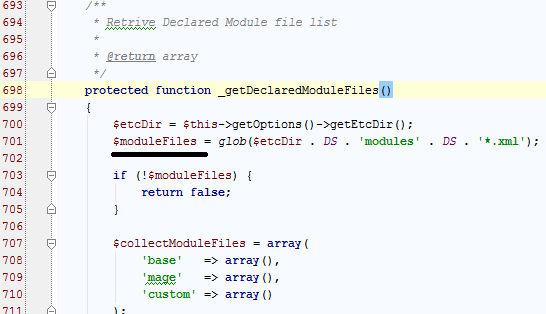
If you install magento using the downloader you might have not have the local folder. No problem go ahead and create it now, this represents code pools. What are code pools for? It’s magentos way to compact functionality and organize modules. A core pool is for basic functionality provided by the magento team. A community pool is where you put modules created by third parties. You can get such modules from magento connect. The official online marketplace for magento extensions. If you are developing modules for magento community we recommend you do so in community folder.

In all other cases we recommend use the local pool. For example you can put all your custom modules in a local folder. There are two additional benefits the magento achieves by putting a code in a code pools.

The first is the ability for you to easily install magento upgrades.The second is to overwrite core functionality by prioritization skin. This prioritization is the most interesting for us as magento developers. How does magento achieve this second benefit? The code pools receive the priority with a local pool having the highest priority and the core pool the lowest. Code in the lib folder has the lower priority than all three of the code pools. Let’s take a look inside Mage.php with logic resides, it’s in the app folder. Take a closer look at line 38 to 48. These lines deal with code pools priority. By the way lines 27 to 29 define some short head DS replaces the longer php constant directory separator PS path separator, BP represent the essence the path to the root folder.

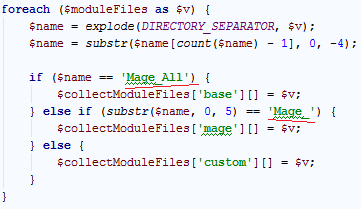
16:30

\app\code\core\Mage\Core\Model\Config.php



F.1.

Then of our names from the app/etc/modules folder are loaded into an array and sorted according to a sequence



F.2.

This sequence is based on the names of the files we want to get core configuration file names first a community and local configuration file names lacks the resulting array will have found names with the prefix mage underscore all (Mage\_All) ..

then of our names with the prefix Mage

and finally the rest to the configuration file names (f.2. последняя строка)



F.3.

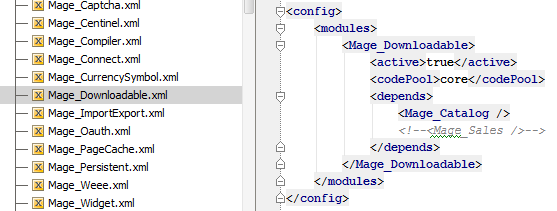
Next, magento to reach the content the files in this array into a configuration object

And sorts it according to the dependencies ($uncortedConfig->extend($fileConfig))

this object won’t contain much information

as we already know the configuration files in the modules folder are just initial configuration files

let’s look at one of them



F.4.

Mage\_Downloadable.xml

They specify the activity state: <active>true</active>

Code pool <codePool>core</codePool>

And dependencies for the module so once magento finishes this step

Will have a configuration object listing all the modules in the ordered their dependencies after that



F.5.

The system iterates each module in an object in if the module in an object in if the modules active flag is set to true reach the modules etc/config.xml file that is this system merges all the data from these files

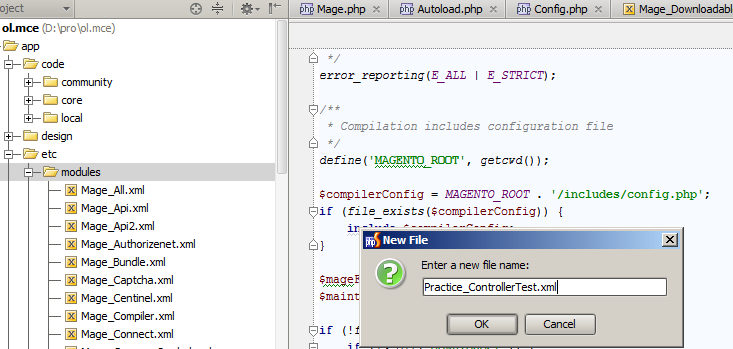
Into this subject not know the order in which magento loads configuration files

If ($mergeModel->loadFile($configFile)){

$mergeToObject->extend($mergeModel, true);

}

It’s time to create the initial configuration file

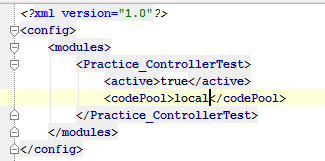


F.6.

Practice underscore controller test dot xml the app/etc/modules folder

And at the content I show in the slide (и содержание я покажу на слайде)

Let’s take a close look at these lines (f.7.)



f.7.

the note configures like a global container of our entire configuration

as I said before the system merges all configuration files from different places

that’s why each configuration file start its data from the config node

the same works for node modules

the modules node is a container for all modules we have a magento

we must specify here our new module so the system could find and run it

and we’ve done that in this line

you remember the underscore naming conventions we discussed earlier

accordingly we specify the module using the pattern namespace underscore module name

(Namespace\_ModuleName)

The next two lines to find the activity state and codePool in the module

These lines mean that when the system loads the modules configure xml

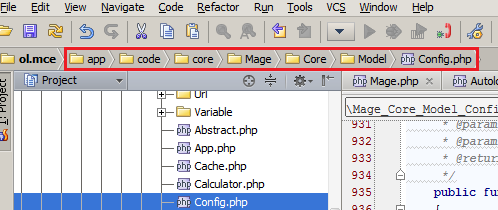
Will check if active is true q.v.:f.5. (if ($module->is(‘active’)){…}

Enough so will build the path as follows: app/code/Practice/ControllerTest/etc/config.xml

You can see how the system composes the path in the file

App code core Mage Core Model Config.php (\app\code\core\Mage\Core\Model\Config.php)

q.v. f.8.



F.8.

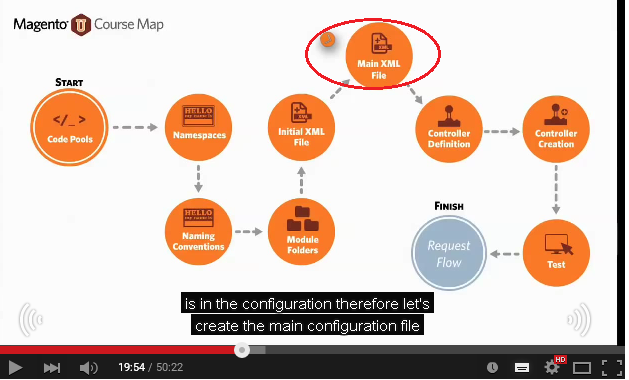
On line 907 (ol 956): $configFile = $this->getModuleDir(‘etc’, $modNmae).DS.$fileName;

Under creating the main configuration files

We need to specify somewhere their module has a controller

The only place to do that is in the configuration therefore let’s create the main configuration file

(q.v. f.9.)

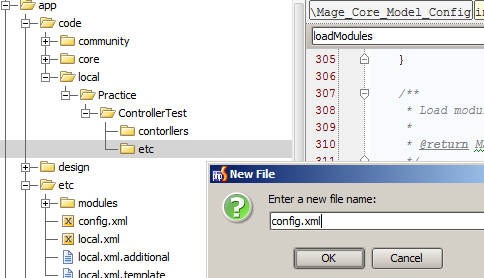
v 

f.9.

config.xml, by the way the name config.xml is hardcoded do not use another name

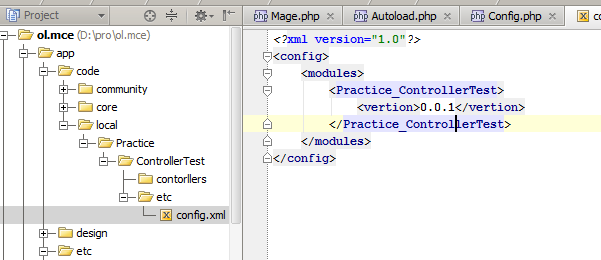
the full path is app/code/local/Practice/ControllerTest/etc/config.xml

the minimum information we must add are the lines between config text



f.10.

let’s walk through each line (q.v. f.11.)



f.11.

we can see here several familiar lines:

config, modules, and Practice\_ControllerTest

when the system merges the initial Practice\_ControllerTest on xml

and this config.xml our configuration object will contain the following combined data.

**Combined Data**

<Practice\_ControllerTest>

<version>0.0.1</version>

<active>true</active>

<codePool>local</codePool>

</Practice\_ControllerTest>

Получается файл: app\etc\modules\Practice\_ControllerTest.xml

И app\code\local\Practice\ControllerTest\etc\config.xml

Соединили!!

…

The config.xml file can be as large as you need

When a module is deactivated that is the note active is set to false not if its additional configuration

Files are loaded

Reducing the size of the configuration object

this is one of the benefits of splitting the configuration

especially since the second piece at the configuration is often several xml files. Ok.

Back to the content config.xml in the modules

at sea folder why do we had a version number here

taking advantage of this convention is a handy way upgrade your module

when you include the version number

the system checks to see if there were any database upgrade scripts to run

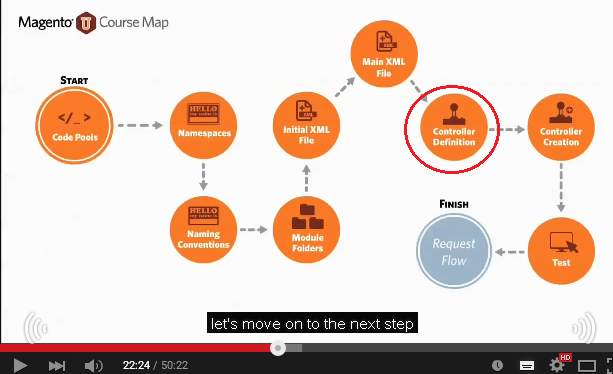
will talk in detail about this later

strictly speaking we don’t need a version number for this particular module

because we don’t have any tables in a database yet

still its good coding style to specify a version for your module

let’s move on to the next step



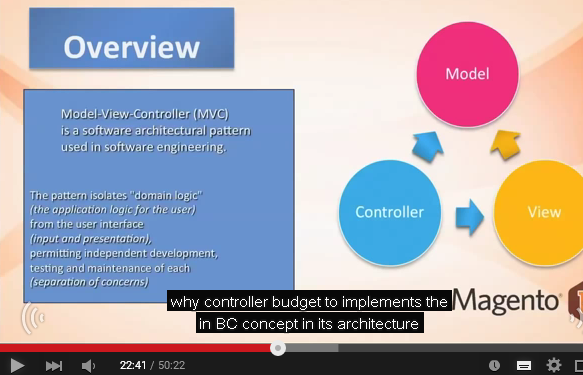
f.12.

defining the controller the next code we want to add

is a definition for our controller especially since the controller is the only working something in this course

why a controller?

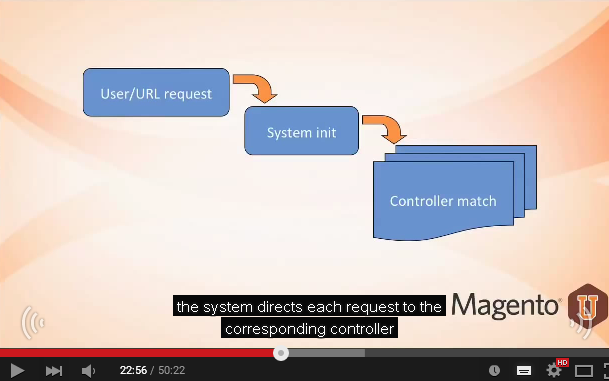
Magento to implements the in MVC concept in its architecture



f.13.

MVC (Model-View-Controller) standard for Model View Controller

We assume you know the details so the MVC concept



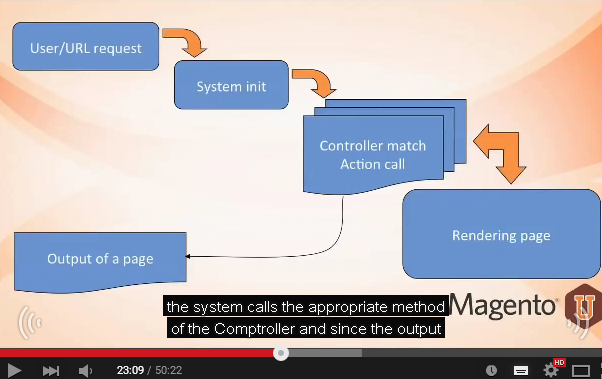
F.14.

The system directs each request to the corresponding controller

And other words the system searches for controller

Which corresponds to the requested path to the URL

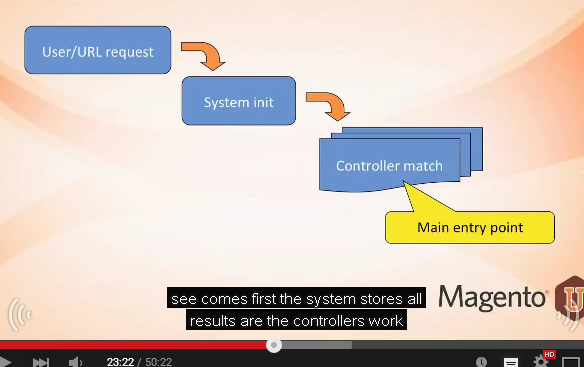
Then the system calls the appropriate method of the Controller and since the output to the client



F.15.

An interesting detail was that in the MVC acronym

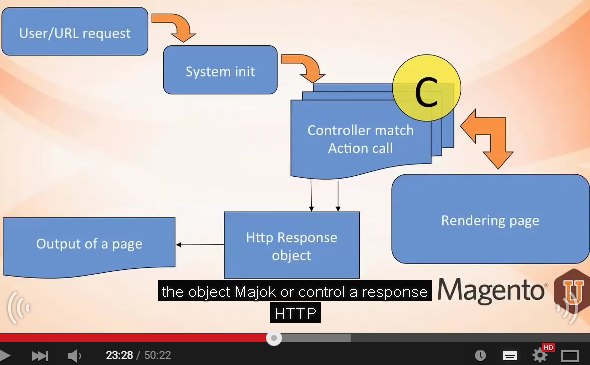
See(возможено это буква “C” = Controller) comes at the (app?) but in magento request processing



f.16.

see (C) comes first

the system stores all results are the controllers work

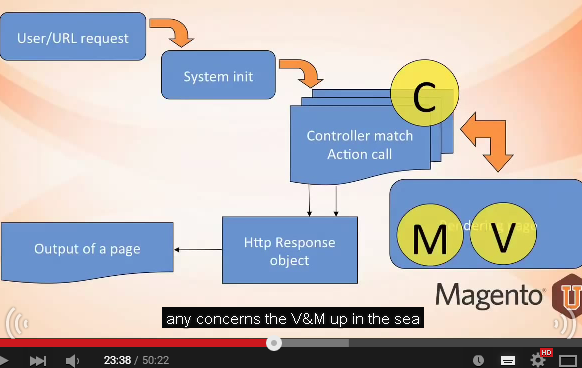


F.17.

The object Mage Core or controller response HTTP

We call this the rendering process

Any concerns the V&M and MVC (em-vi-ci) (q.v.: f.18.)

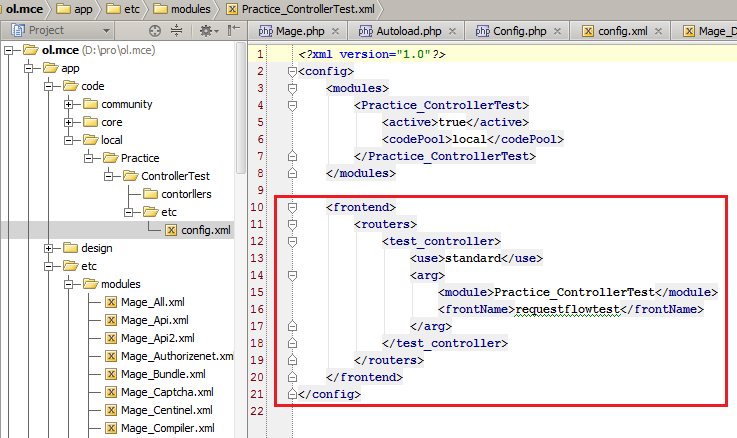


f.18.

in this course we focus on the C have an acronym

these lines inform the system about our controller

go ahead and add these lines to your config.xml file (q.v.: f.19.)



f.19.

you may wonder what frontend, routers, test\_controller in other lines here mean

briefly frontend is an area in the area in the router (or routers) node

also pronounce router

we create the configuration for the corresponding router

containing our route in controller

we used this controller as the unique identifiers

wrapping all over controller definition

<test\_controller>

…

What are area router and route?

Speaking a various we have the backend to administer the system

Area = backend (admin), frontend (visitors)

And the front inside for visitors

These two are areas

We call them frontend and admin its appear concept

If some options must be applied to all areas we use the global area

The router is in essence a class which is given a router role

This class is responsible for gathering all routes in corresponding controller classes

Which were addressed to a (or you are?)

The route is a concept we use to create the URL

In mostly concern the concern the frontName node

When the system processes request the system searches for corresponding controller

<frontName>requestflowtest</frontName>

Trying to find a match among all the frontName values and the loaded system configuration

And a route name from the current request

That is why we call it around because the system uses it to arrive at a corresponding controller

Will go into more detail later in the request flow video

Let’s create the last piece the controller file (q.v.: f.1.20.)

Now will the controller we defined in the previous lesson



f.1.20.

open the app/code/local/Practice/ContorllerTest/controllers folder

create a new PHP file called IndexController.php

|  |  |
| --- | --- |
|  | Notice the upper case “Controller”.  We’ll discuss this further in a later lesson. |

f.1.21.

open the file and type the code you see on the slide

what is a controller’s role in the system

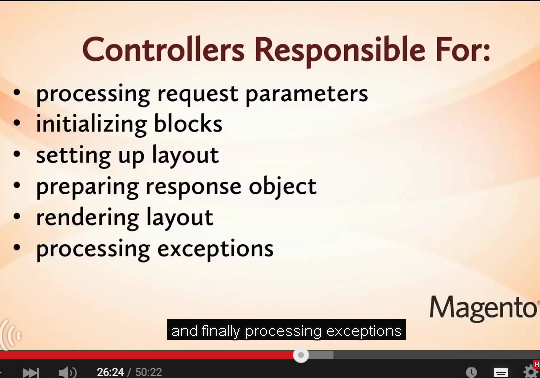
controllers are responsible for processing the request parameters

initializing blocks setting up the layout

preparing the response object

rendering the layout

and finally processing exceptions (f.1.22.)

****

f.1.22.

some of these responsibilities are optional but whichever ones you need put them in the controller

a common mistake is to use business logic in the controller

don’t do it

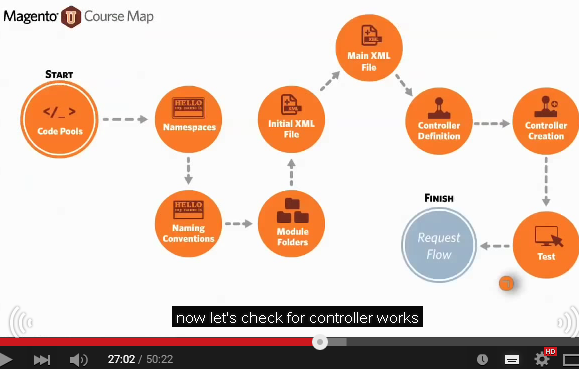
the exception is if you working with the module involving very simple business logic

earlier when we created the at sea(C) and controllers folder for module

you may have noticed the model and helper folders have other modules

the proper place for business logic is in these two folders a given module

now let’s check for controller works (f.1.23.)



f.1.23.

point (put) your browser to the URL you see on the screen

(<http://magento.loc/magento-1.5.1.0/requestflowtest>)

(my browser: http://ol.mce/ requestflowtest)

Obviously the part of the URL depends on where you located your magento

The easiest way to compose the URL is to point your browser to your home page

Independent slash index to php slash request flow test (писать слитно requestflowtest)

Without anything else

you see hello world yea it works