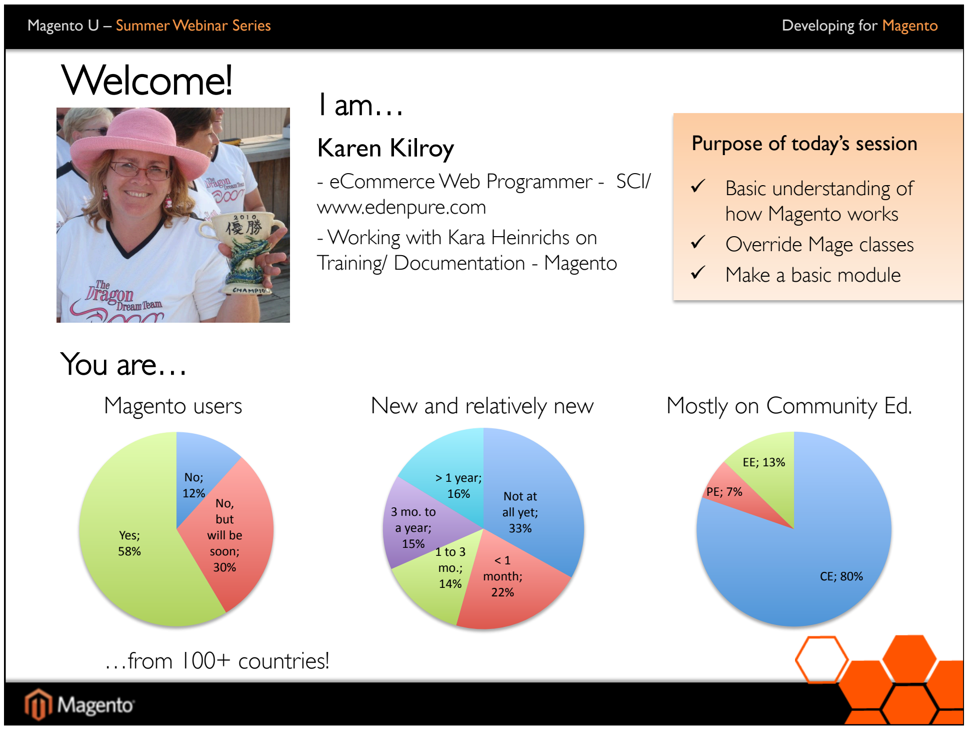
Summer Webinar Series
August 28, 2010

DEVELOPING FOR MAGENTO



Magento U – Summer Webinar Series | Developing for Magento

Welcome!

I am…

Karen Kilroy

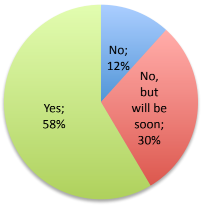
- eCommerce Web Programmer - SCI/ www.edenpure.com
- Working with Kara Heinrichs on Training/ Documentation - Magento

Purpose of today's session

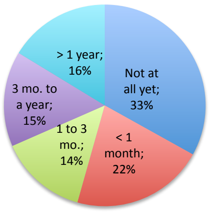
- ✓ Basic understanding of how Magento works
- ✓ Override Mage classes
- ✓ Make a basic module

You are…

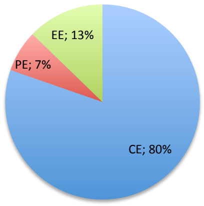
Magento users



New and relatively new

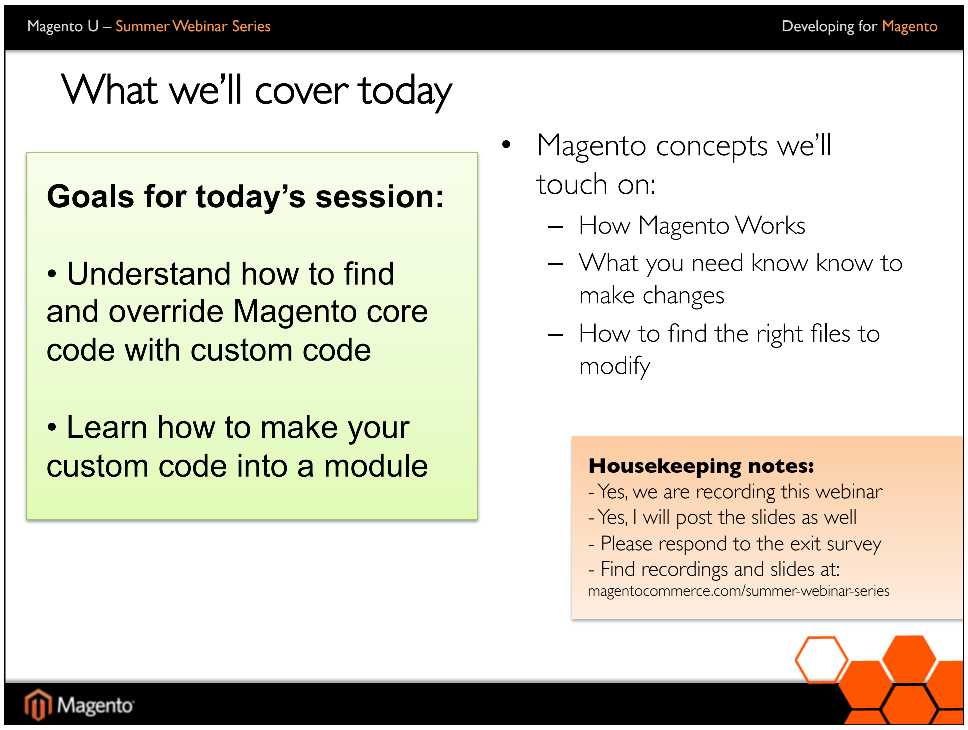


Mostly on Community Ed.



…from 100+ countries!

Magento



Magento U – Summer Webinar Series Developing for Magento

What we'll cover today

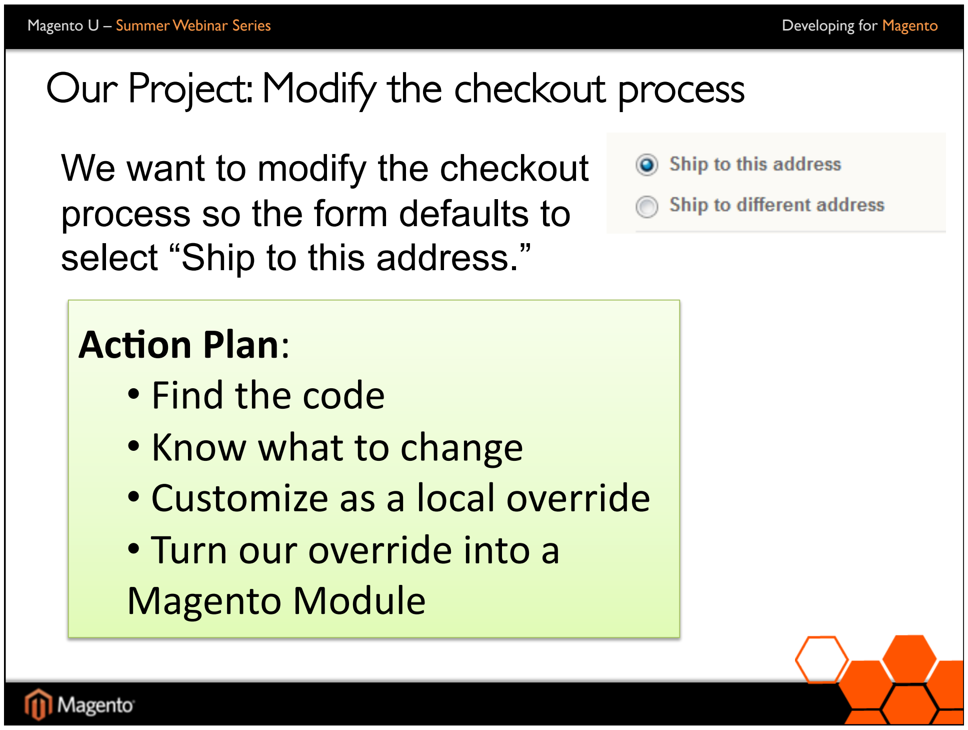
Goals for today's session:

- Understand how to find and override Magento core code with custom code
- Learn how to make your custom code into a module

- Magento concepts we'll touch on:
  - How Magento Works
  - What you need know know to make changes
  - How to find the right files to modify

Housekeeping notes:
- Yes, we are recording this webinar
- Yes, I will post the slides as well
- Please respond to the exit survey
- Find recordings and slides at: magentocommerce.com/summer-webinar-series

Magento



Magento U – Summer Webinar Series Developing for Magento

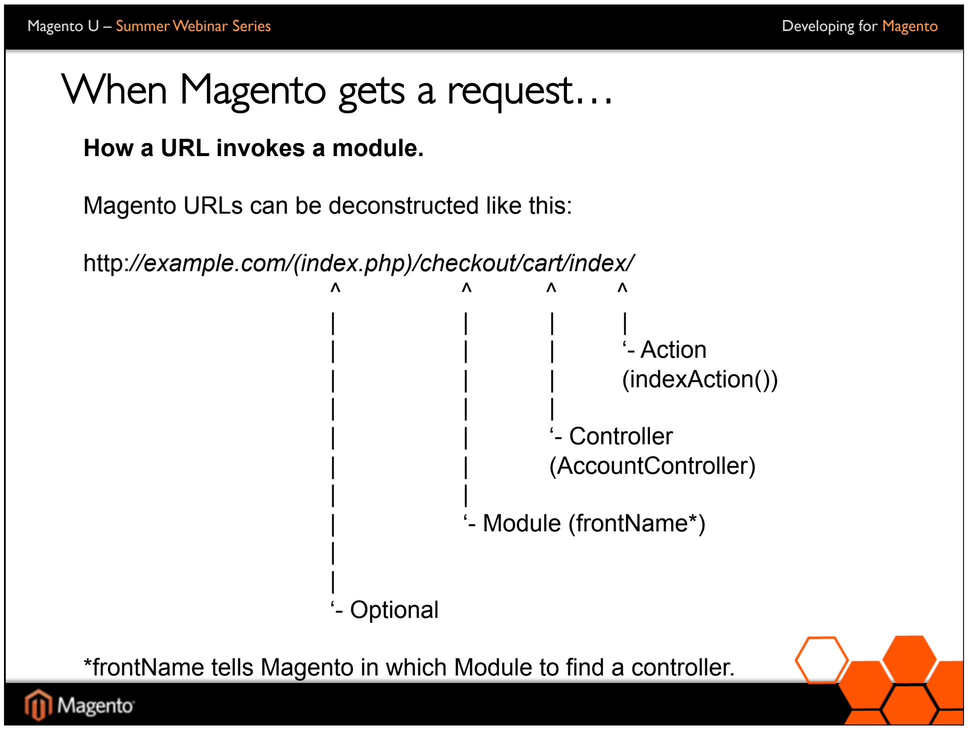
Our Project: Modify the checkout process

We want to modify the checkout process so the form defaults to select "Ship to this address."

[✓ Ship to this address]
Ship to different address

Action Plan:
- Find the code
- Know what to change
- Customize as a local override
- Turn our override into a Magento Module

Magento



When Magento gets a request…

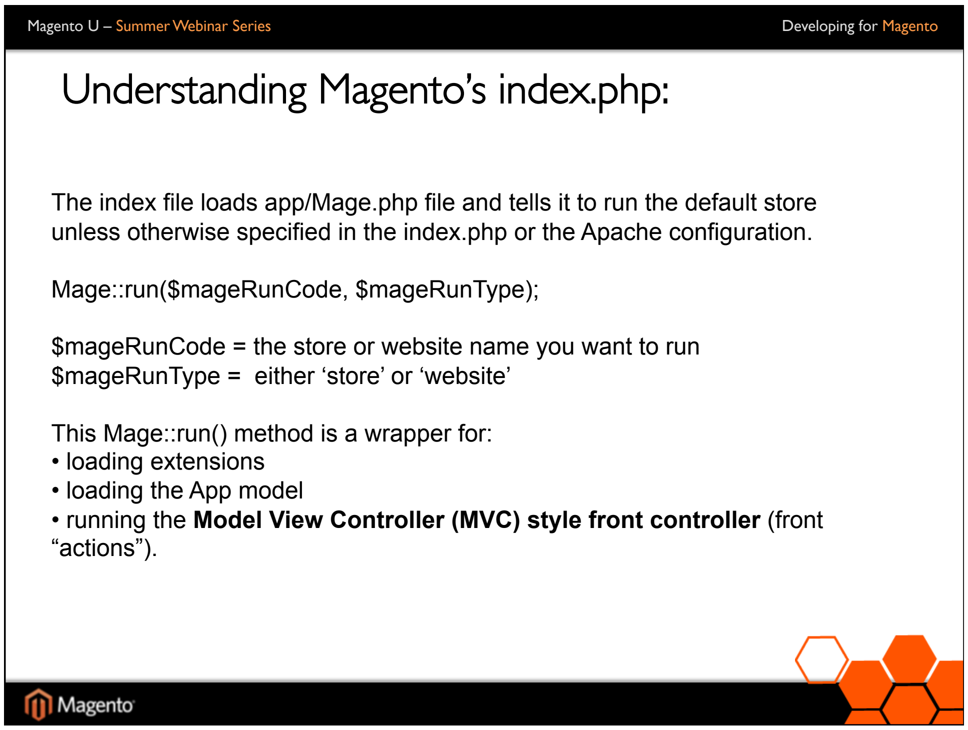
How a URL invokes a module.

Magento URLs can be deconstructed like this:

```
http://example.com/(index.php)/checkout/cart/index/
                        ^           ^      ^    ^
                        |           |      |    |
                        |           |      |    '- Action
                        |           |      |    (indexAction())
                        |           |      |
                        |           |      '- Controller
                        |           |      (AccountController)
                        |           |
                        |           '- Module (frontName*)
                        |
                        |
                        '- Optional
```

*frontName tells Magento in which Module to find a controller.

Magento



Understanding Magento's index.php:

The index file loads app/Mage.php file and tells it to run the default store unless otherwise specified in the index.php or the Apache configuration.

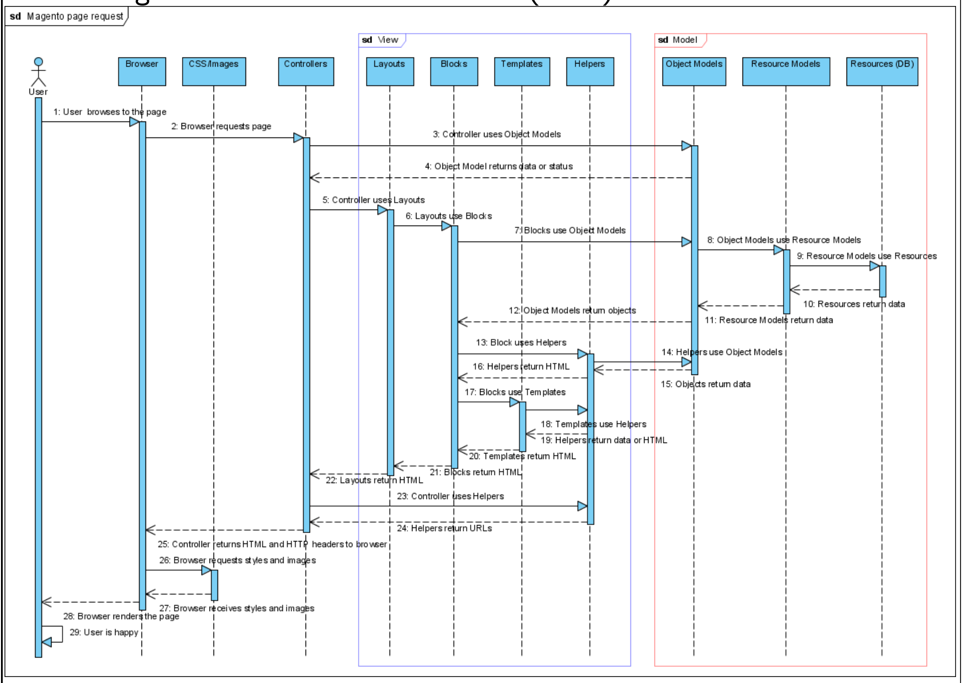
Mage::run($mageRunCode, $mageRunType);

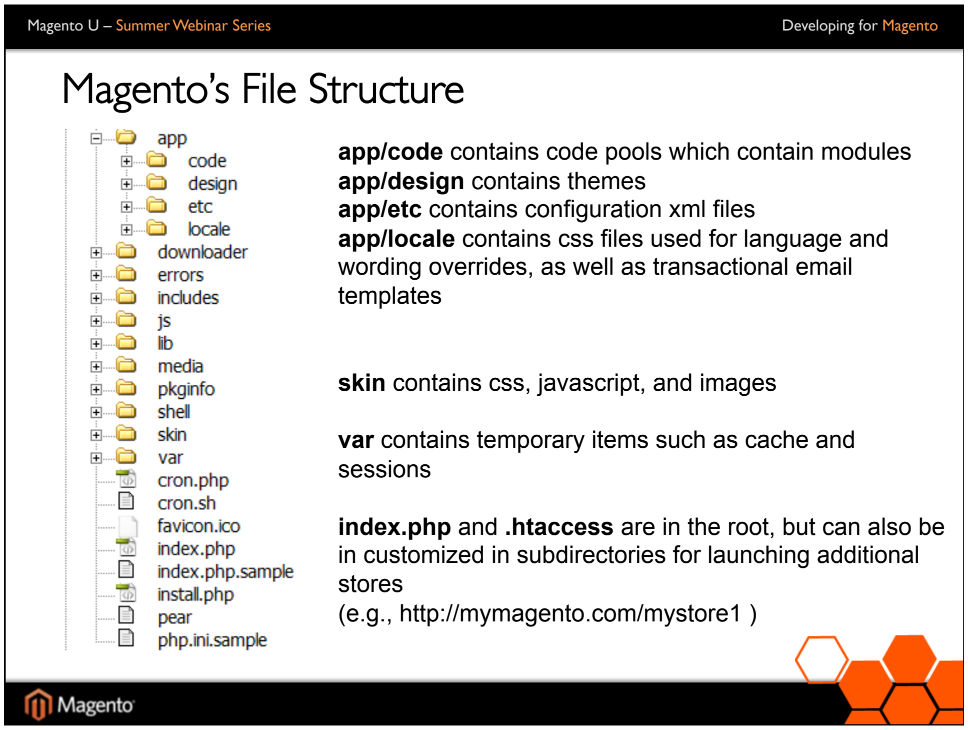
$mageRunCode = the store or website name you want to run
$mageRunType = either 'store' or 'website'

This Mage::run() method is a wrapper for:
- loading extensions
- loading the App model
- running the **Model View Controller (MVC) style front controller** (front "actions").

Magento

Magento's Model View Controller (MVC) Architecture


sd Magento page request
sd View
sd Model
User
Browser
CSS/Images
Controllers
Layouts
Blocks
Templates
Helpers
Object Models
Resource Models
Resources (DB)
1: User browses to the page
2: Browser requests page
3: Controller uses Object Models
4: Object Model returns data or status
5: Controller uses Layouts
6: Layouts use Blocks
7: Blocks use Object Models
8: Object Models use Resource Models
9: Resource Models use Resources
10: Resources return data
11: Resource Models return data
12: Object Models return objects
13: Block uses Helpers
14: Helpers use Object Models
15: Objects return data
16: Helpers return HTML
17: Blocks use Templates
18: Templates use Helpers
19: Helpers return data or HTML
20: Templates return HTML
21: Blocks return HTML
22: Layouts return HTML
23: Controller uses Helpers
24: Helpers return URLs
25: Controller returns HTML and HTTP headers to browser
26: Browser requests styles and images
27: Browser receives styles and images
28: Browser renders the page
29: User is happy



Magento U – Summer Webinar Series Developing for Magento

Magento's File Structure

- app
  - code
  - design
  - etc
  - locale
- downloader
- errors
- includes
- js
- lib
- media
- pkginfo
- shell
- skin
- var
- cron.php
- cron.sh
- favicon.ico
- index.php
- index.php.sample
- install.php
- pear
- php.ini.sample

app/code contains code pools which contain modules
app/design contains themes
app/etc contains configuration xml files
app/locale contains css files used for language and wording overrides, as well as transactional email templates

skin contains css, javascript, and images

var contains temporary items such as cache and sessions

index.php and **.htaccess** are in the root, but can also be in customized in subdirectories for launching additional stores
(e.g., http://mymagento.com/mystore1)

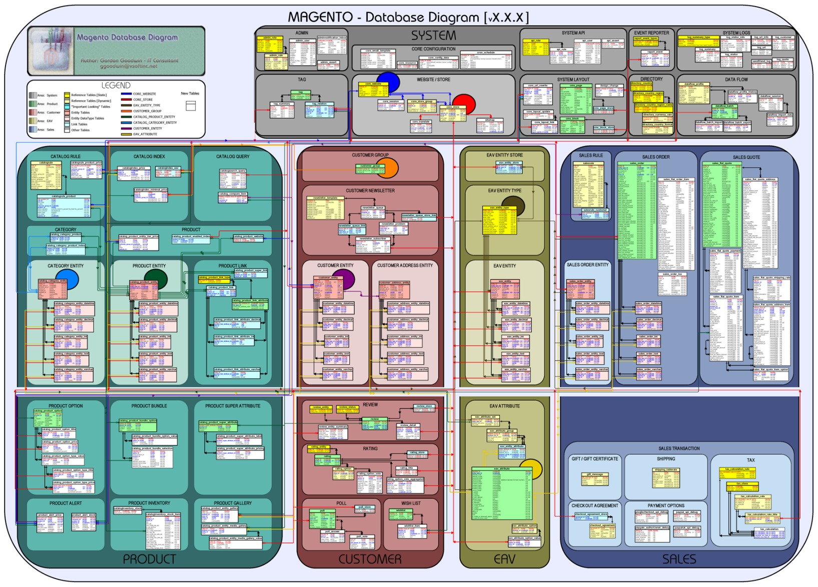
Magento

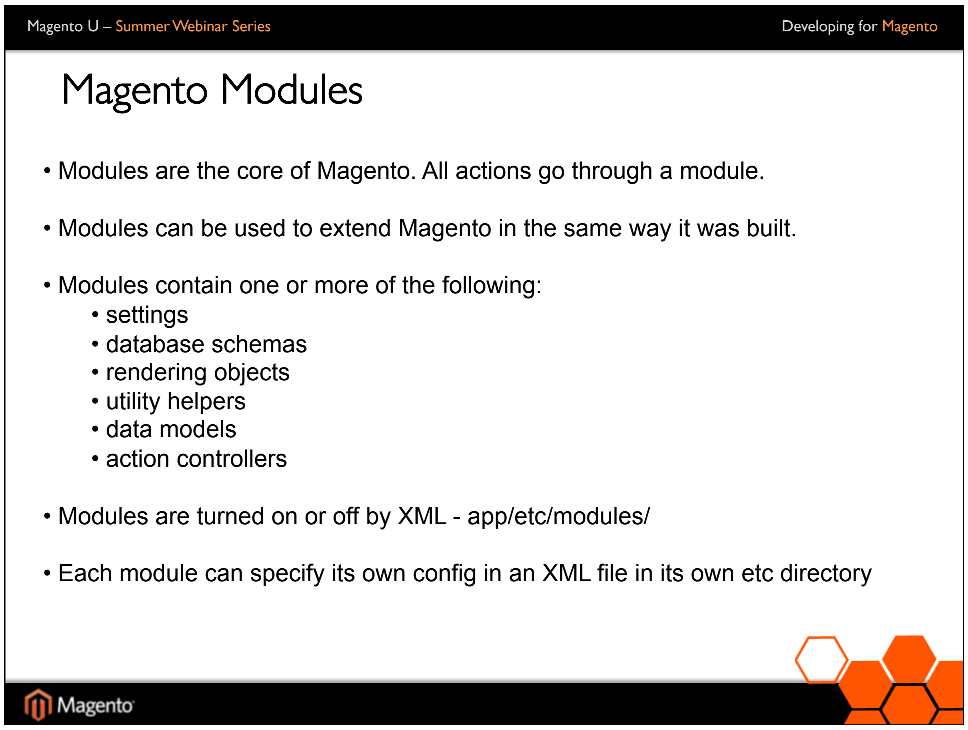
Magento's Template System: Review

- Magento's templates are in the app/design directory
- php and html = phtml files in template directory
- Controlled by a set of XML files, or layout files, in layout directory
- Many things you want to change may be within the template system
- This topic was covered in Magento Themes webinar

Magento's Database Structure

- Entity Attribute Value (EAV) Structure
- Good for extending data structure but not for ad-hoc queries



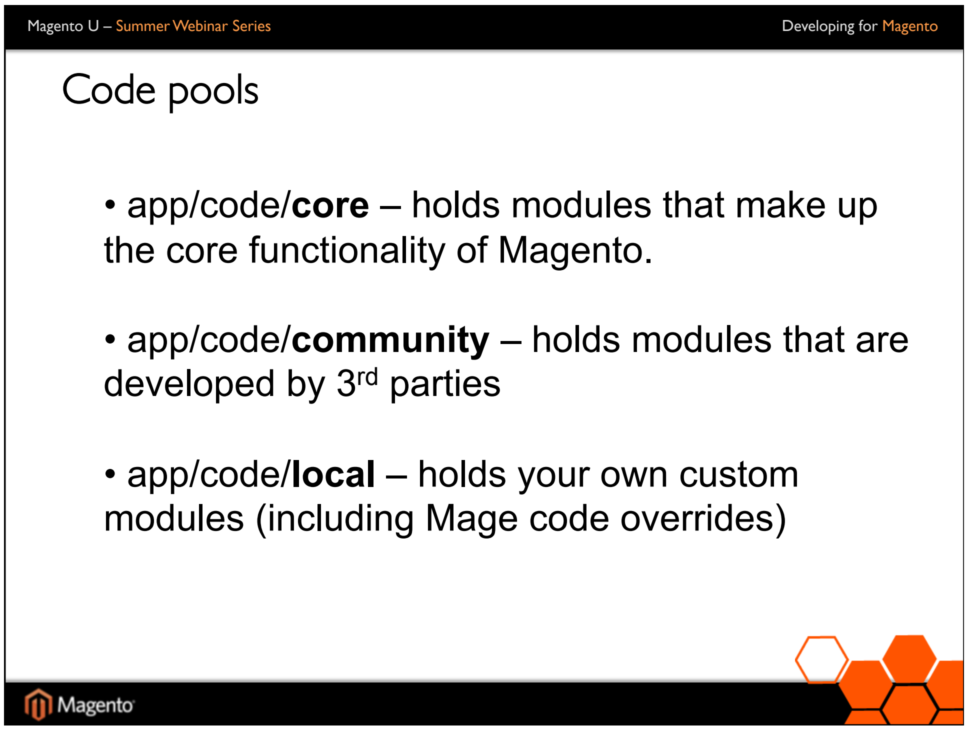


Magento U – Summer Webinar Series Developing for Magento

Magento Modules

- Modules are the core of Magento. All actions go through a module.
- Modules can be used to extend Magento in the same way it was built.
- Modules contain one or more of the following:
  - settings
  - database schemas
  - rendering objects
  - utility helpers
  - data models
  - action controllers
- Modules are turned on or off by XML - app/etc/modules/
- Each module can specify its own config in an XML file in its own etc directory

Magento

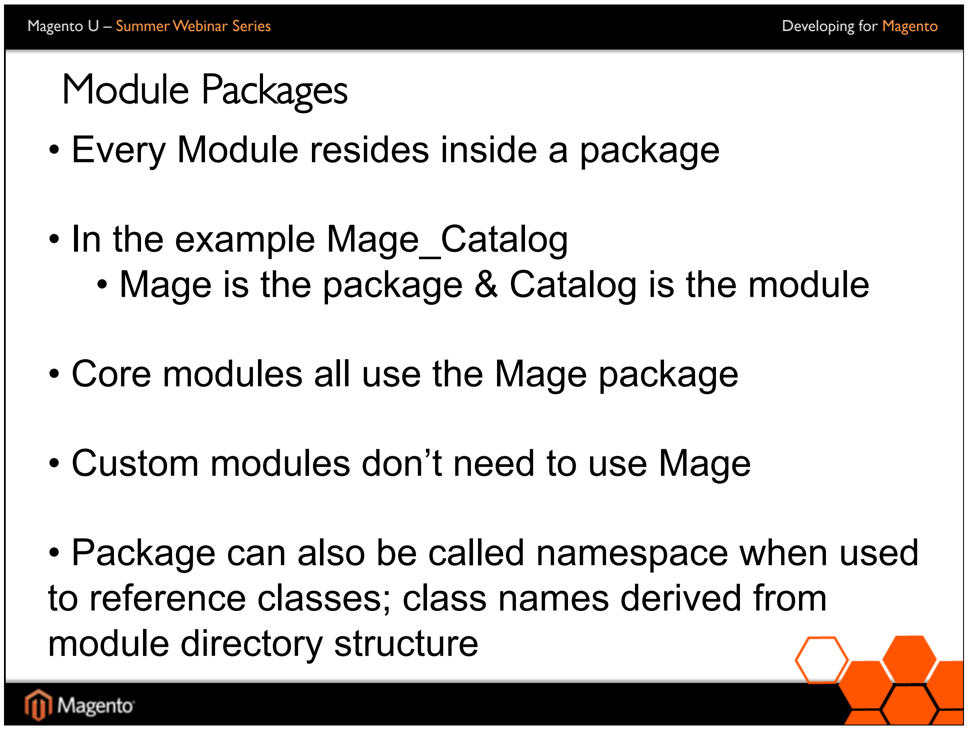


Magento U – Summer Webinar Series Developing for Magento

Code pools

- app/code/**core** – holds modules that make up the core functionality of Magento.
- app/code/**community** – holds modules that are developed by 3rd parties
- app/code/**local** – holds your own custom modules (including Mage code overrides)

Magento

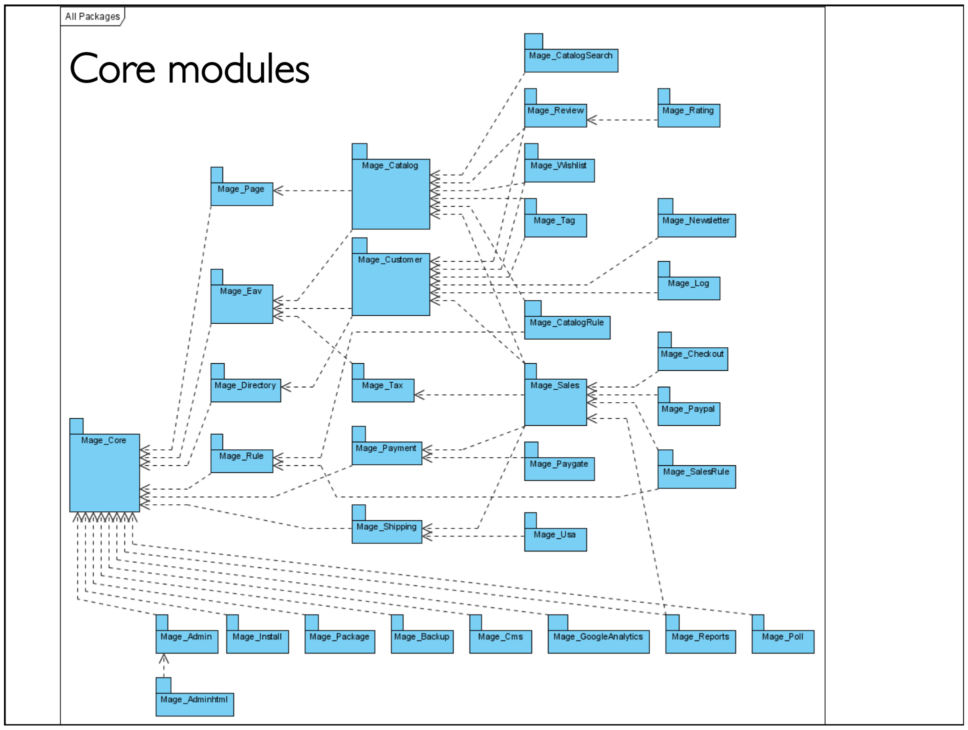


Magento U – Summer Webinar Series Developing for Magento

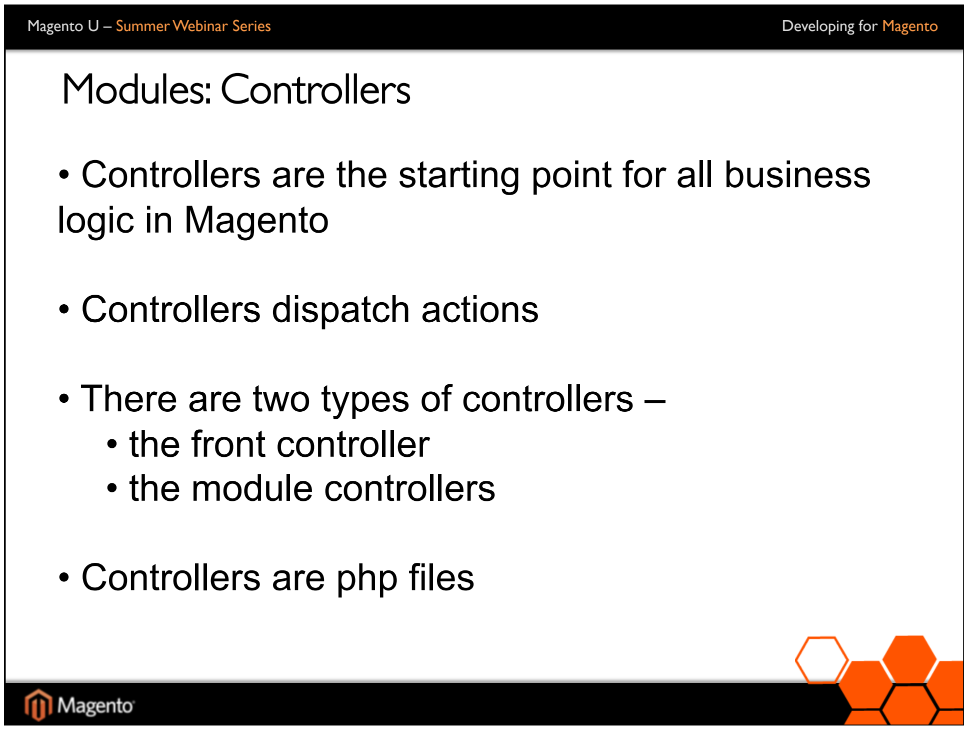
Module Packages

- Every Module resides inside a package
- In the example Mage_Catalog
    - Mage is the package & Catalog is the module
- Core modules all use the Mage package
- Custom modules don't need to use Mage
- Package can also be called namespace when used to reference classes; class names derived from module directory structure

Magento

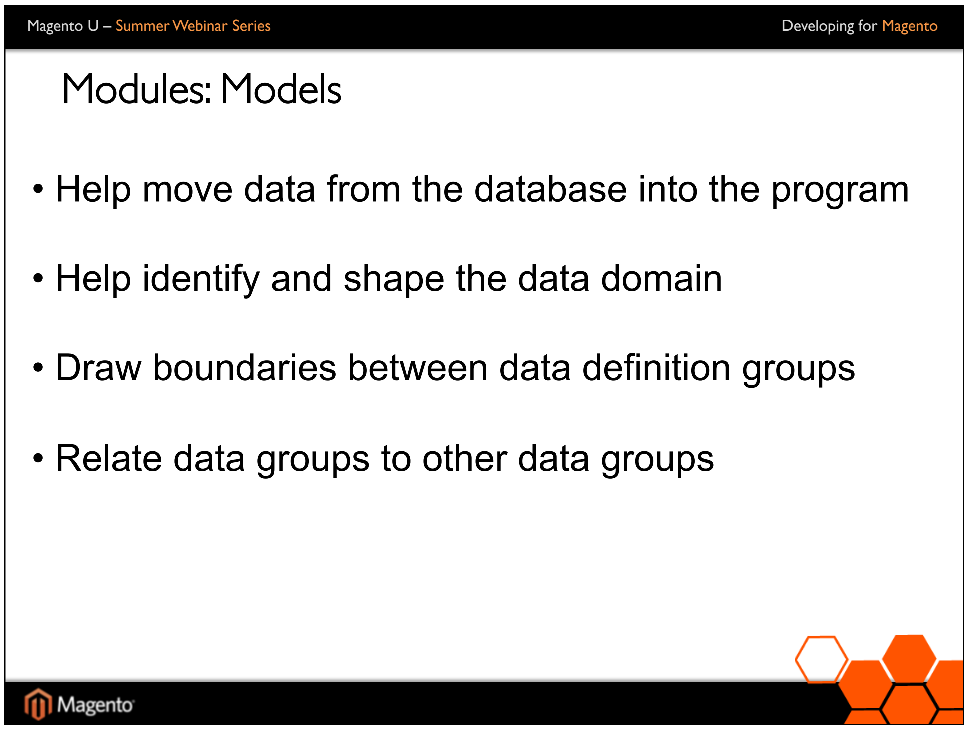


Core modules



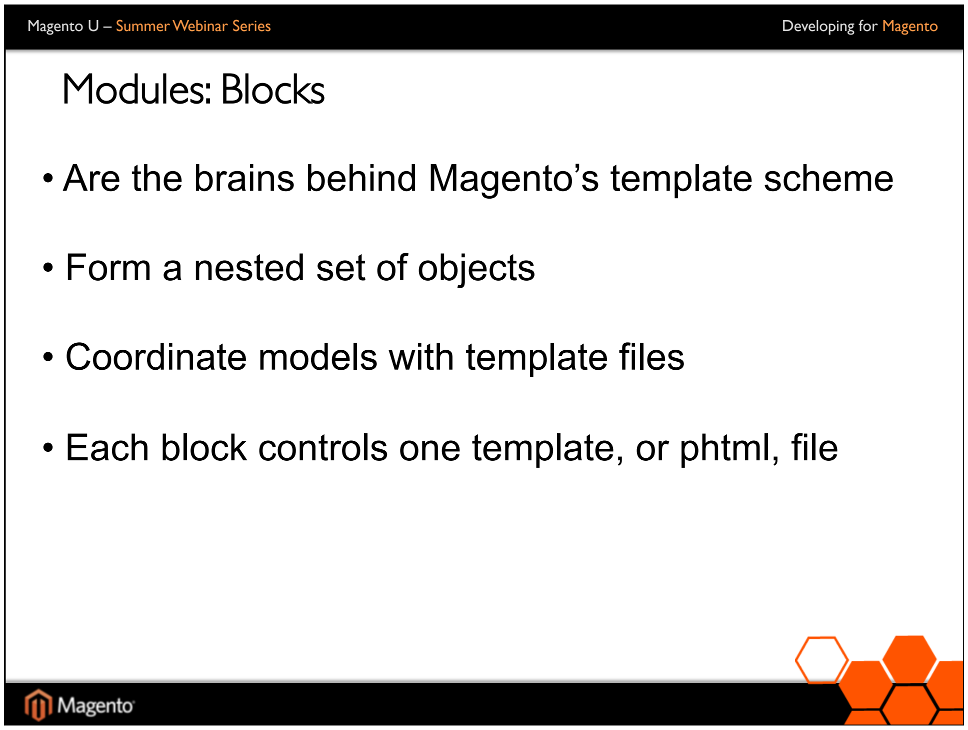
Modules: Controllers

- Controllers are the starting point for all business logic in Magento
- Controllers dispatch actions
- There are two types of controllers –
  - the front controller
  - the module controllers
- Controllers are php files



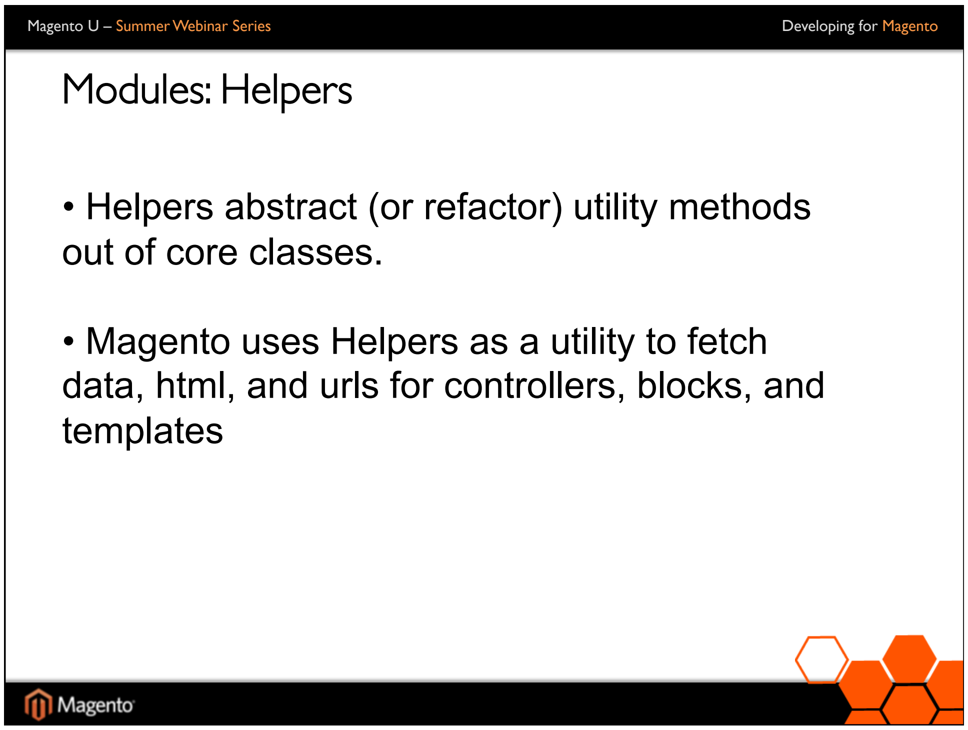
Modules: Models

- Help move data from the database into the program
- Help identify and shape the data domain
- Draw boundaries between data definition groups
- Relate data groups to other data groups



Modules: Blocks

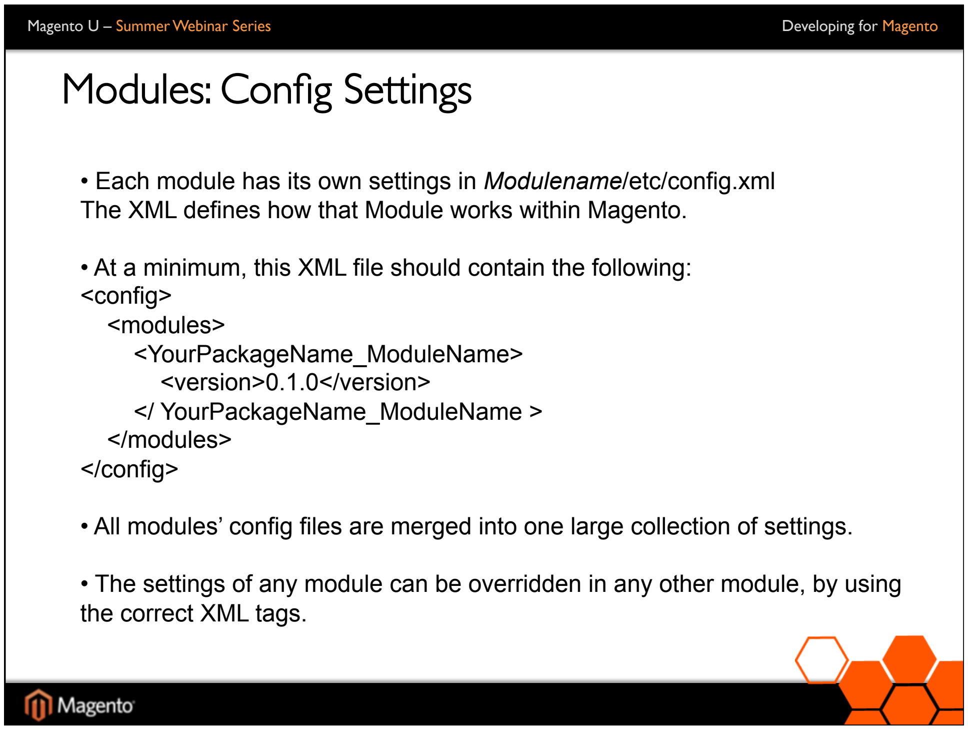
- Are the brains behind Magento's template scheme
- Form a nested set of objects
- Coordinate models with template files
- Each block controls one template, or phtml, file



Modules: Helpers

• Helpers abstract (or refactor) utility methods out of core classes.

• Magento uses Helpers as a utility to fetch data, html, and urls for controllers, blocks, and templates



Modules: Config Settings

• Each module has its own settings in *Modulename*/etc/config.xml
The XML defines how that Module works within Magento.

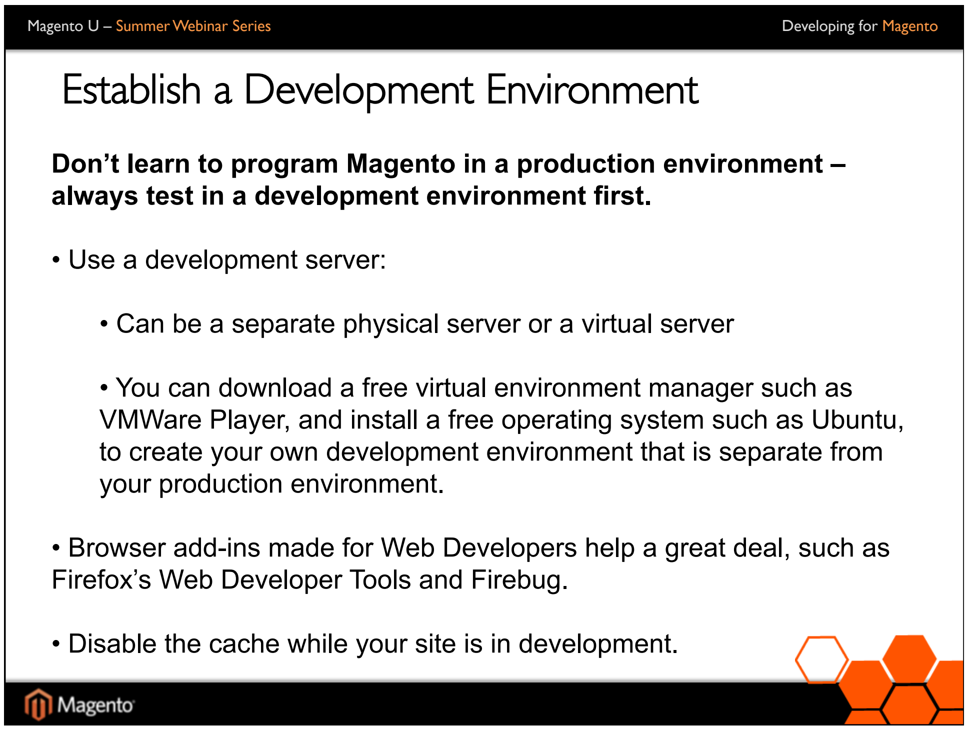
• At a minimum, this XML file should contain the following:

```
<config>
   <modules>
      <YourPackageName_ModuleName>
         <version>0.1.0</version>
      </ YourPackageName_ModuleName >
   </modules>
</config>
```

• All modules' config files are merged into one large collection of settings.

• The settings of any module can be overridden in any other module, by using the correct XML tags.

Magento



Establish a Development Environment

Don't learn to program Magento in a production environment – always test in a development environment first.

• Use a development server:

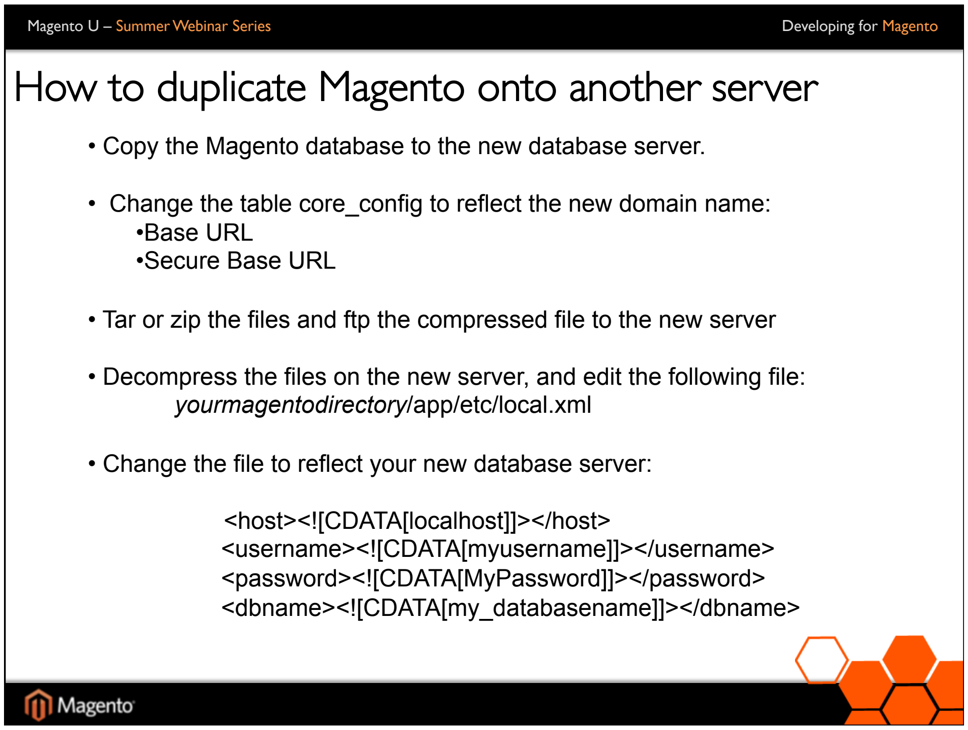
  • Can be a separate physical server or a virtual server

  • You can download a free virtual environment manager such as VMWare Player, and install a free operating system such as Ubuntu, to create your own development environment that is separate from your production environment.

• Browser add-ins made for Web Developers help a great deal, such as Firefox's Web Developer Tools and Firebug.

• Disable the cache while your site is in development.

Magento

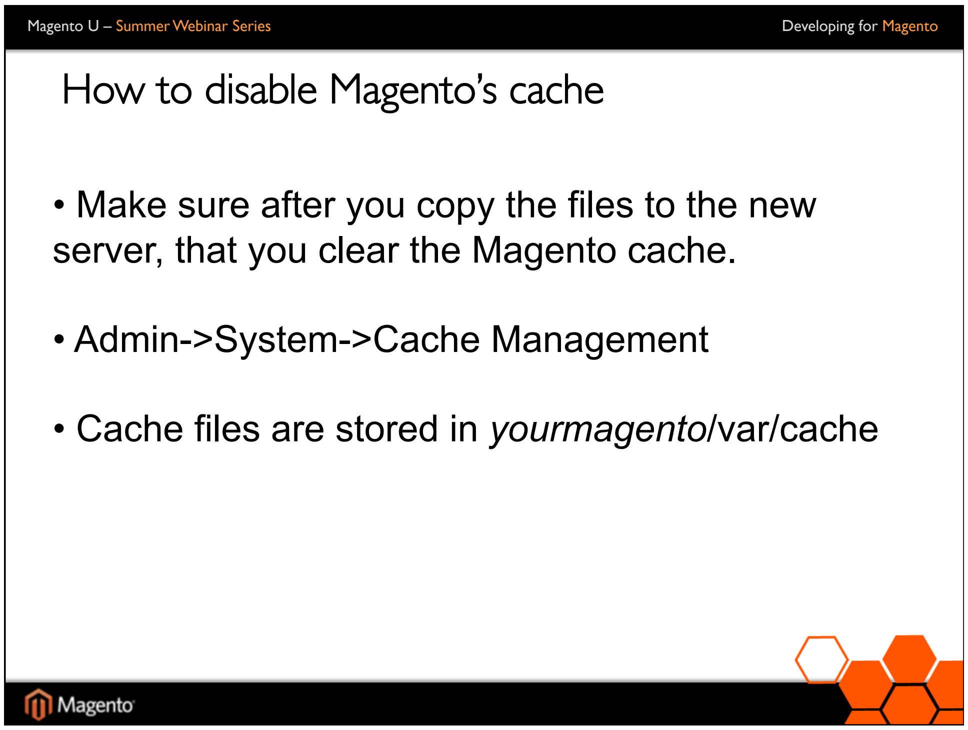


How to duplicate Magento onto another server

- Copy the Magento database to the new database server.
- Change the table core_config to reflect the new domain name:
  - Base URL
  - Secure Base URL
- Tar or zip the files and ftp the compressed file to the new server
- Decompress the files on the new server, and edit the following file:
  yourmagentodirectory/app/etc/local.xml
- Change the file to reflect your new database server:

```
<host><![CDATA[localhost]]></host>
<username><![CDATA[myusername]]></username>
<password><![CDATA[MyPassword]]></password>
<dbname><![CDATA[my_databasename]]></dbname>
```

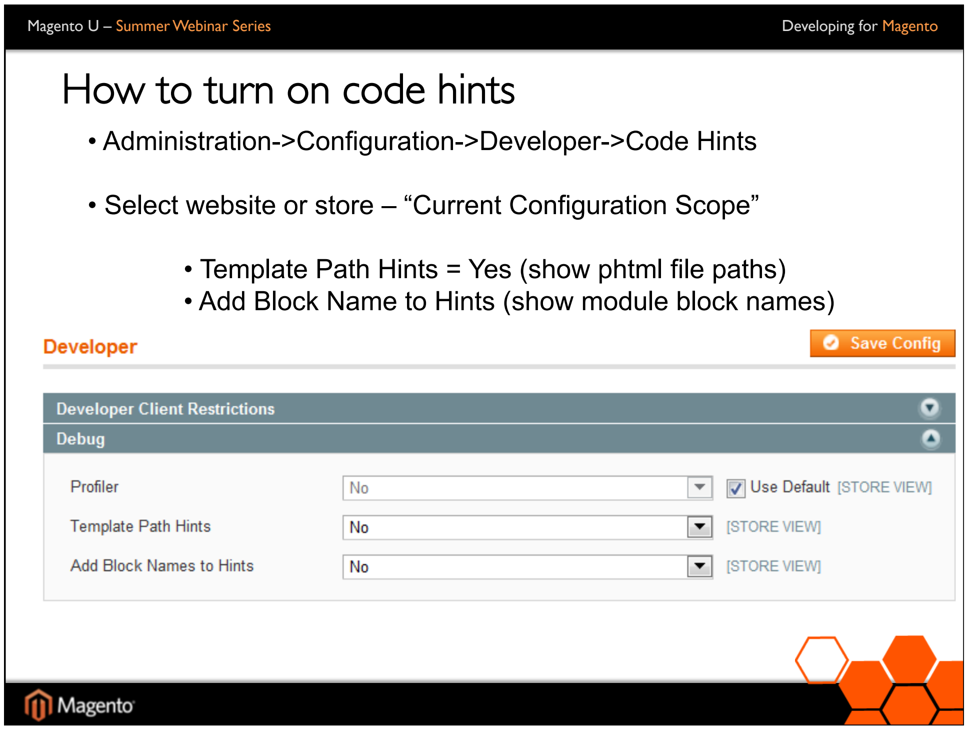
Magento



How to disable Magento’s cache

- Make sure after you copy the files to the new server, that you clear the Magento cache.
- Admin->System->Cache Management
- Cache files are stored in *yourmagento*/var/cache

Magento



Magento U – Summer Webinar Series Developing for Magento

How to turn on code hints

- Administration->Configuration->Developer->Code Hints
- Select website or store – “Current Configuration Scope”
    - Template Path Hints = Yes (show phtml file paths)
    - Add Block Name to Hints (show module block names)

Developer

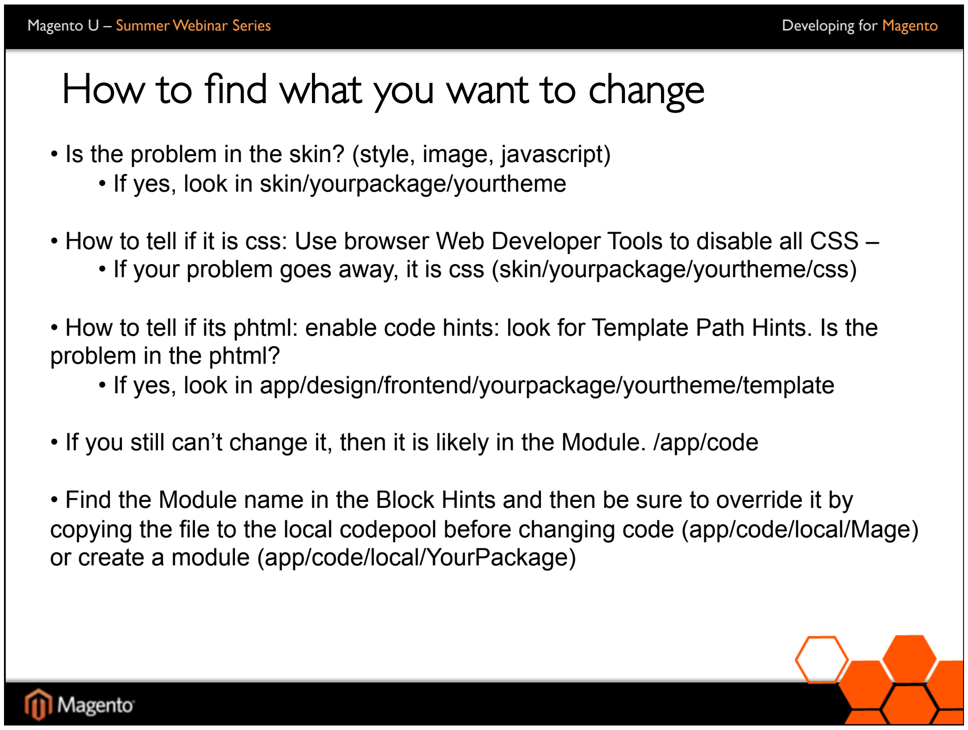
Save Config

Developer Client Restrictions

Debug

Profiler	No	✓ Use Default	[STORE VIEW]
Template Path Hints	No		[STORE VIEW]
Add Block Names to Hints	No		[STORE VIEW]

Magento

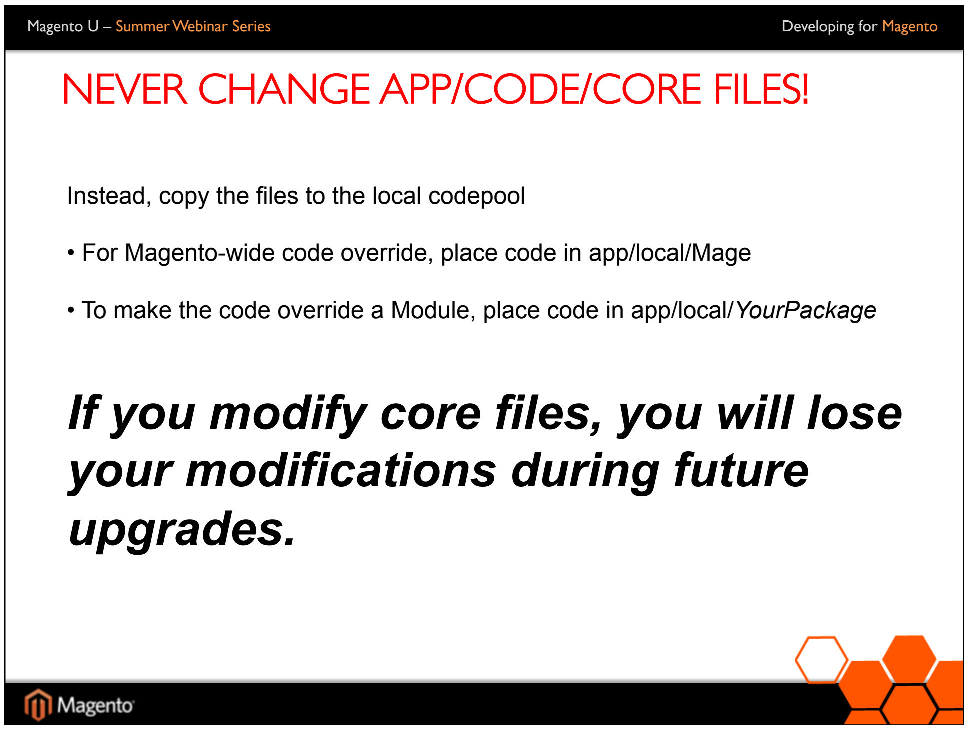


Magento U – Summer Webinar Series Developing for Magento

How to find what you want to change

- Is the problem in the skin? (style, image, javascript)
    - If yes, look in skin/yourpackage/yourtheme
- How to tell if it is css: Use browser Web Developer Tools to disable all CSS –
    - If your problem goes away, it is css (skin/yourpackage/yourtheme/css)
- How to tell if its phtml: enable code hints: look for Template Path Hints. Is the problem in the phtml?
    - If yes, look in app/design/frontend/yourpackage/yourtheme/template
- If you still can’t change it, then it is likely in the Module. /app/code
- Find the Module name in the Block Hints and then be sure to override it by copying the file to the local codepool before changing code (app/code/local/Mage) or create a module (app/code/local/YourPackage)

Magento



Magento U – Summer Webinar Series Developing for Magento

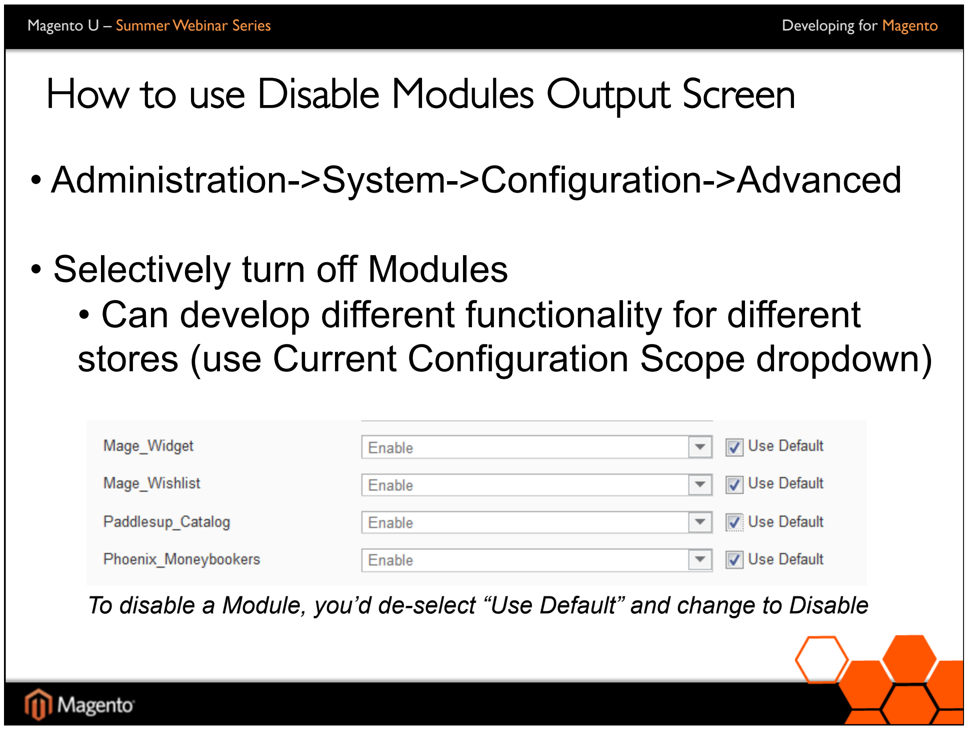
NEVER CHANGE APP/CODE/CORE FILES!

Instead, copy the files to the local codepool

- For Magento-wide code override, place code in app/local/Mage
- To make the code override a Module, place code in app/local/*YourPackage*

If you modify core files, you will lose your modifications during future upgrades.

Magento



Magento U – Summer Webinar Series Developing for Magento

How to use Disable Modules Output Screen

- Administration->System->Configuration->Advanced
- Selectively turn off Modules
  - Can develop different functionality for different stores (use Current Configuration Scope dropdown)

Mage_Widget	Enable	✓ Use Default
Mage_Wishlist	Enable	✓ Use Default
Paddlesup_Catalog	Enable	✓ Use Default
Phoenix_Moneybookers	Enable	✓ Use Default

To disable a Module, you'd de-select "Use Default" and change to Disable

Magento

Project: Set Default - "Ship to this Address"

Step 1:Turn on Code Hints to find the file that needs to be modified

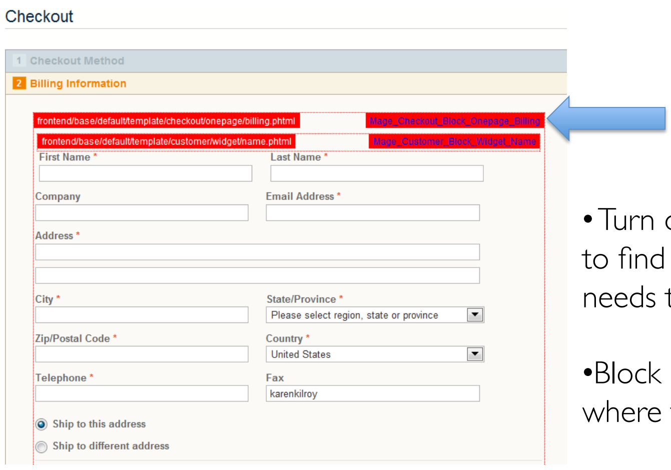
Step 2: Copy the file to the local code pool

Step 3: Modify the code in the local code pool to select "Ship to this Address" as the new default

Step 4: Upload the code and check for results

Project: Set Default - "Ship to this address"

Step 1 – Find the code you want to change


Checkout

1 Checkout Method

2 Billing Information

frontend/base/default/template/checkout/onepage/billing.phtml Mage_Checkout_Block_Onepage_Billing

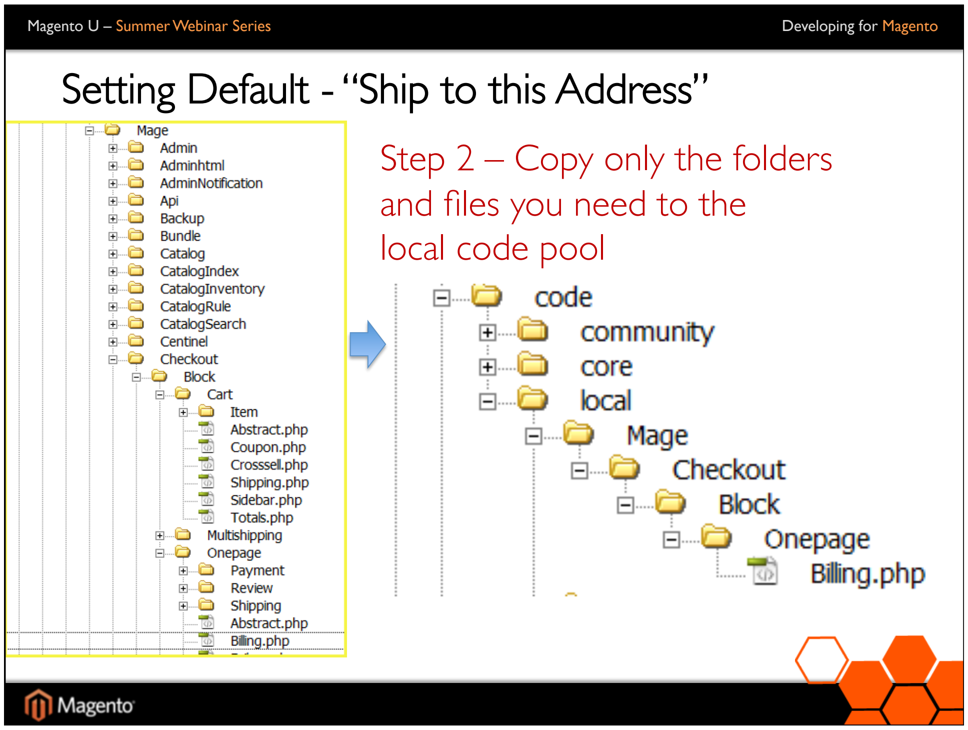
frontend/base/default/template/customer/widget/name.phtml Mage_Customer_Block_Widget_Name

First Name *
Last Name *
Company
Email Address *
Address *
City *
State/Province * Please select region, state or province
Zip/Postal Code *
Country * United States
Telephone *
Fax karenkilroy

● Ship to this address
○ Ship to different address

- Turn on Code Hints to find the file that needs to be modified
- Block hint tells you where to look for file

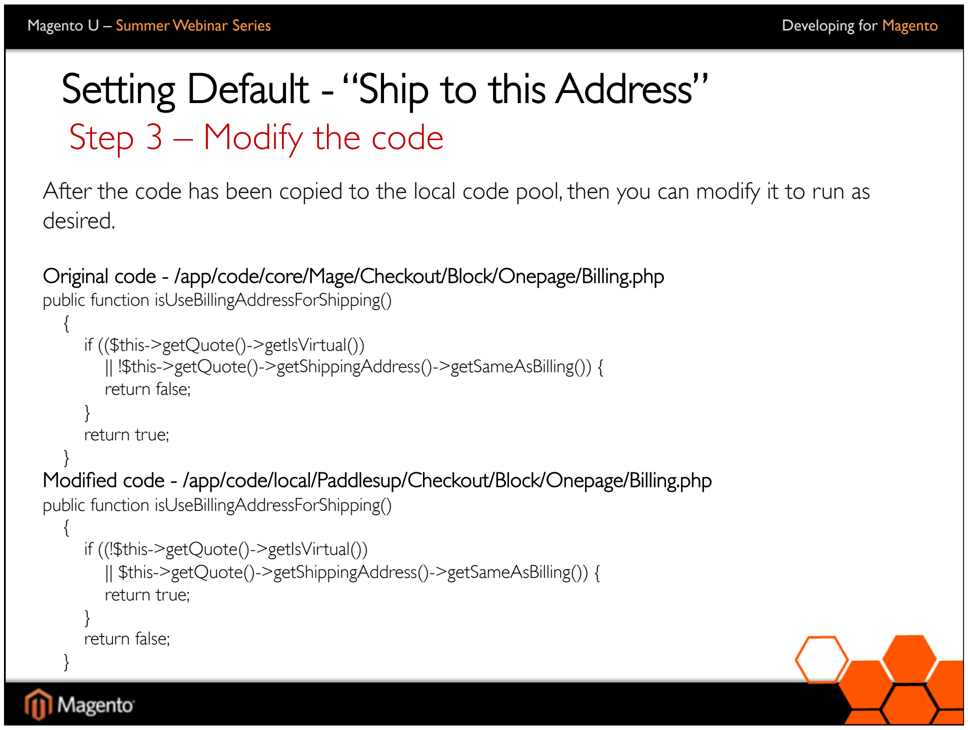
Mage_Checkout_Block_Onepage_Billing = /Mage/Checkout/Onepage/Block/Billing.php



Magento U – Summer Webinar Series Developing for Magento

Setting Default - "Ship to this Address"

Step 2 – Copy only the folders and files you need to the local code pool



Magento U – Summer Webinar Series Developing for Magento

Setting Default - "Ship to this Address"

Step 3 – Modify the code

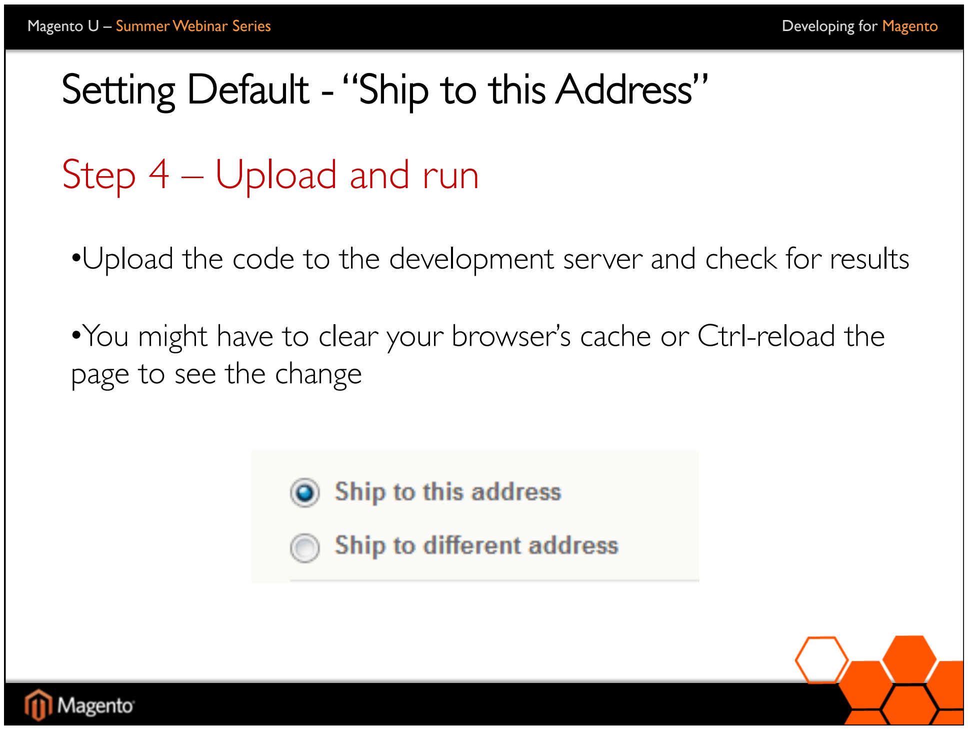
After the code has been copied to the local code pool, then you can modify it to run as desired.

Original code - /app/code/core/Mage/Checkout/Block/Onepage/Billing.php

```
public function isUseBillingAddressForShipping()
    {
        if (($this->getQuote()->getIsVirtual())
            || !$this->getQuote()->getShippingAddress()->getSameAsBilling()) {
            return false;
        }
        return true;
    }
```

Modified code - /app/code/local/Paddlesup/Checkout/Block/Onepage/Billing.php

```
public function isUseBillingAddressForShipping()
    {
        if ((!$this->getQuote()->getIsVirtual())
            || $this->getQuote()->getShippingAddress()->getSameAsBilling()) {
            return true;
        }
        return false;
    }
```



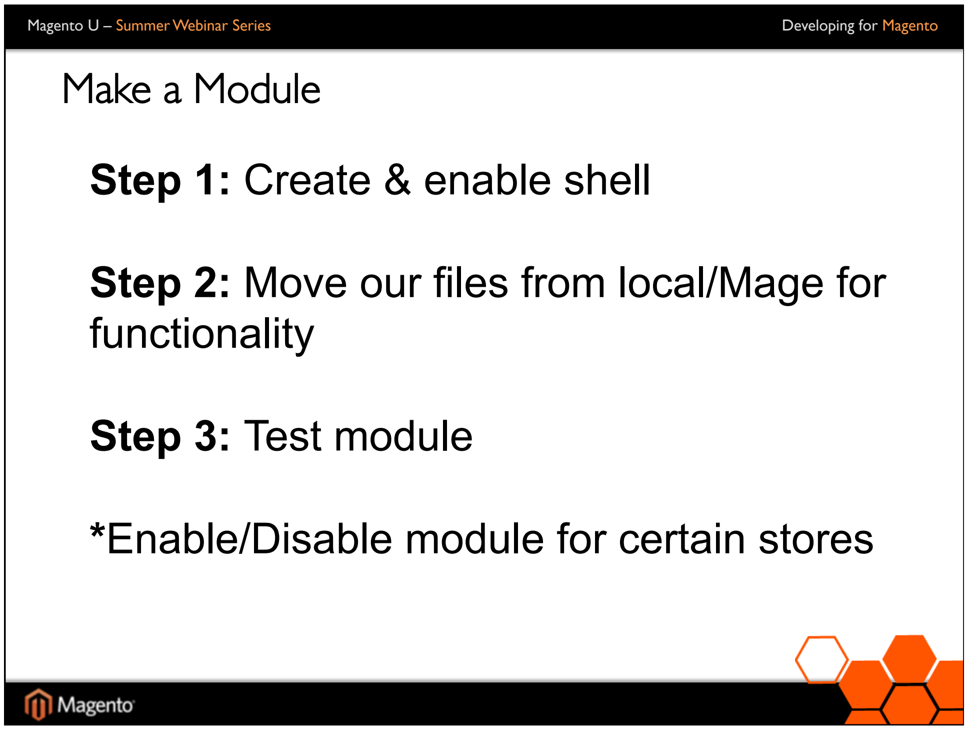
Setting Default - "Ship to this Address"

Step 4 – Upload and run

•Upload the code to the development server and check for results

•You might have to clear your browser's cache or Ctrl-reload the page to see the change

[✓ Ship to this address]
Ship to different address



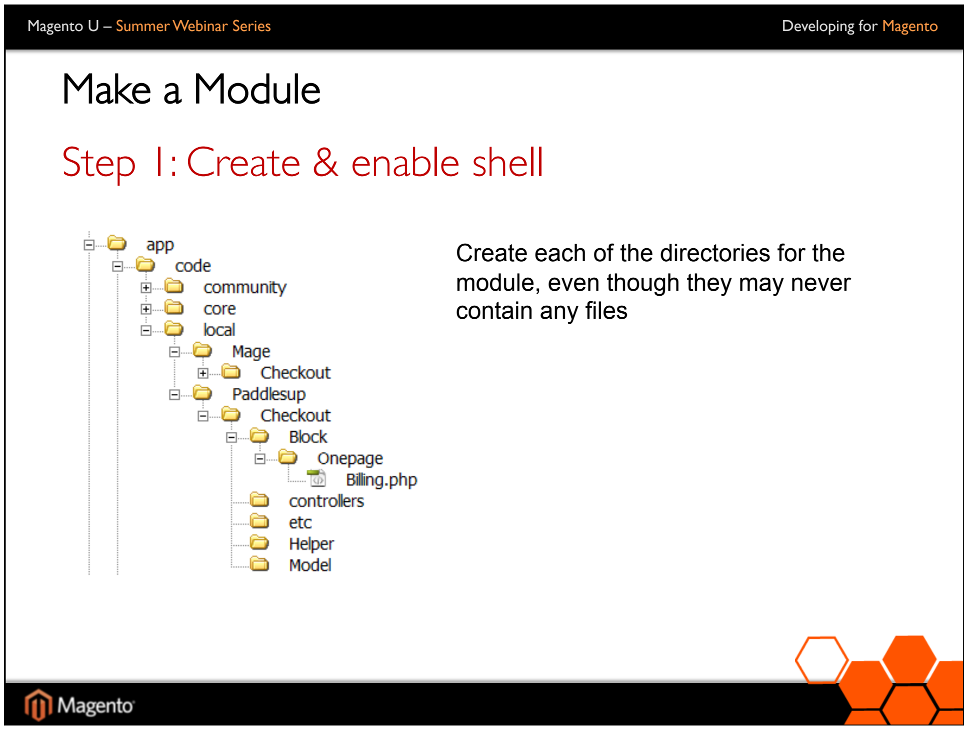
Make a Module

Step 1: Create & enable shell

Step 2: Move our files from local/Mage for functionality

Step 3: Test module

*Enable/Disable module for certain stores



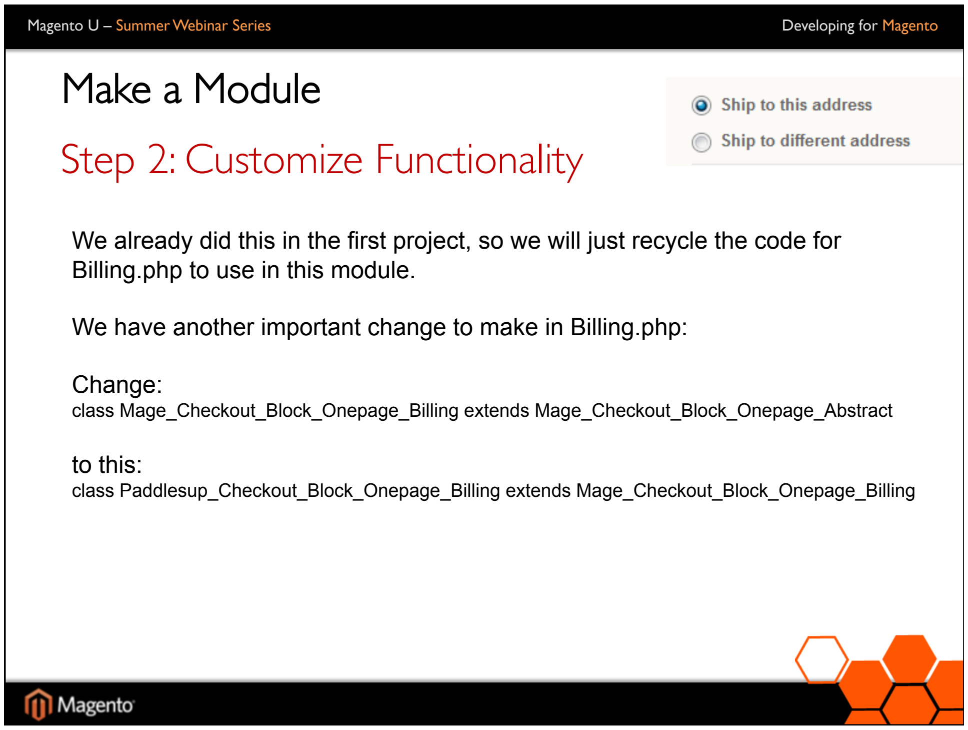
Make a Module

Step 1: Create & enable shell

```
app
  code
    community
    core
    local
      Mage
        Checkout
      Paddlesup
        Checkout
          Block
            Onepage
              Billing.php
          controllers
          etc
          Helper
          Model
```

Create each of the directories for the module, even though they may never contain any files

Magento



Make a Module

Step 2: Customize Functionality

[✓ Ship to this address]
Ship to different address

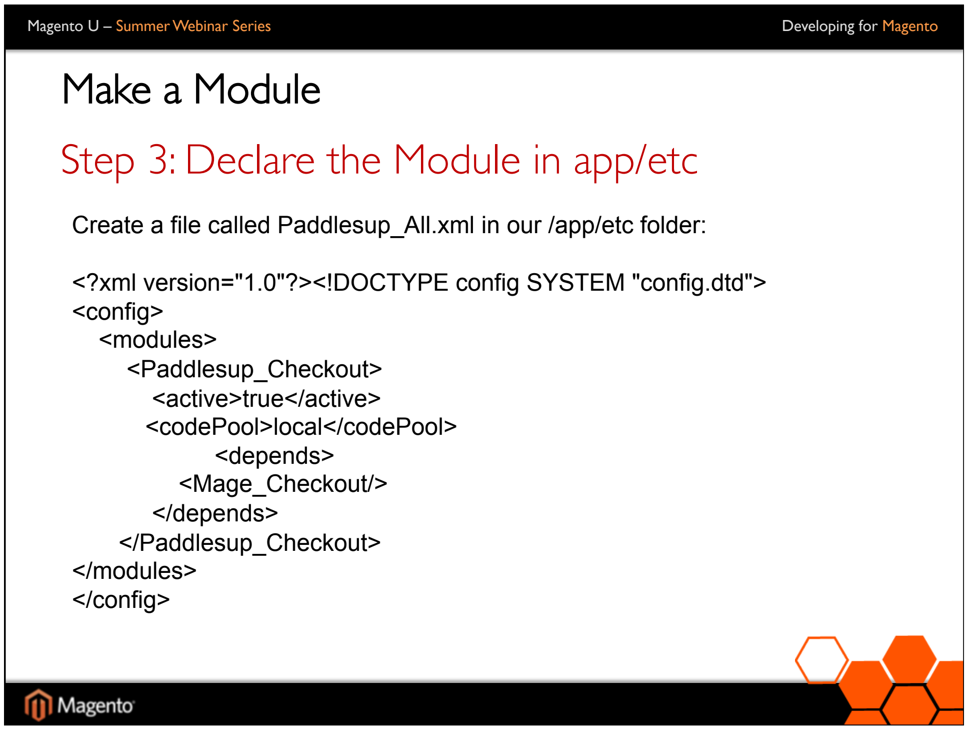
We already did this in the first project, so we will just recycle the code for Billing.php to use in this module.

We have another important change to make in Billing.php:

Change:
class Mage_Checkout_Block_Onepage_Billing extends Mage_Checkout_Block_Onepage_Abstract

to this:
class Paddlesup_Checkout_Block_Onepage_Billing extends Mage_Checkout_Block_Onepage_Billing

Magento


Magento U – Summer Webinar Series Developing for Magento

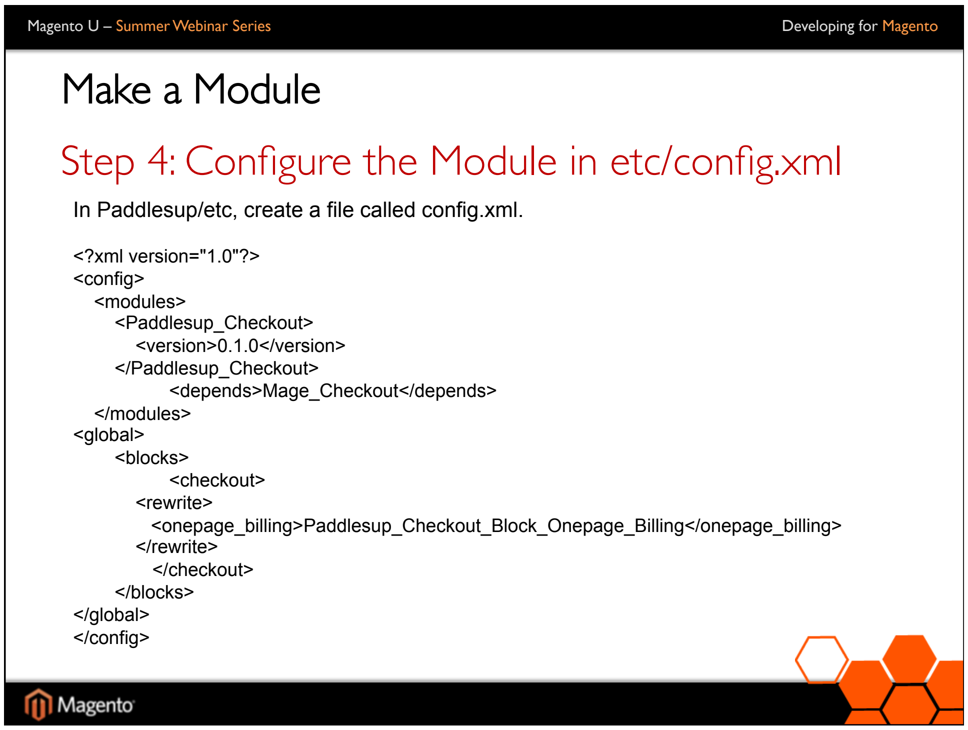
Make a Module

Step 3: Declare the Module in app/etc

Create a file called Paddlesup_All.xml in our /app/etc folder:

```
<?xml version="1.0"?><!DOCTYPE config SYSTEM "config.dtd">
<config>
   <modules>
      <Paddlesup_Checkout>
         <active>true</active>
        <codePool>local</codePool>
               <depends>
           <Mage_Checkout/>
         </depends>
     </Paddlesup_Checkout>
</modules>
</config>
```

Magento


Magento U – Summer Webinar Series Developing for Magento

Make a Module

Step 4: Configure the Module in etc/config.xml

In Paddlesup/etc, create a file called config.xml.

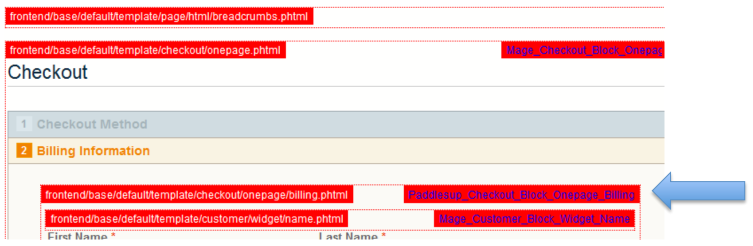
```
<?xml version="1.0"?>
<config>
   <modules>
      <Paddlesup_Checkout>
         <version>0.1.0</version>
      </Paddlesup_Checkout>
            <depends>Mage_Checkout</depends>
   </modules>
<global>
      <blocks>
            <checkout>
         <rewrite>
           <onepage_billing>Paddlesup_Checkout_Block_Onepage_Billing</onepage_billing>
         </rewrite>
           </checkout>
      </blocks>
</global>
</config>
```

Magento

Make a Module

Step 5: Upload & Run

Upload your new module to the server. Check the Configuration->Advanced to make sure your module is enabled. Check code hints to see if it is overriding the core module.



```
frontend/base/default/template/page/html/breadcrumbs.phtml
frontend/base/default/template/checkout/onepage.phtml          Mage_Checkout_Block_Onepag
Checkout
1 Checkout Method
2 Billing Information
frontend/base/default/template/checkout/onepage/billing.phtml   Paddlesup_Checkout_Block_Onepage_Billing
frontend/base/default/template/customer/widget/name.phtml       Mage_Customer_Block_Widget_Name
First Name *                                  Last Name *
```

Success: Paddlesup_Checkout_Block_Onepage_Billing shows in code hints.

NOTE: Design elements can also be used with a particular module, yet they will need to reside in a theme.

Magento

Summary

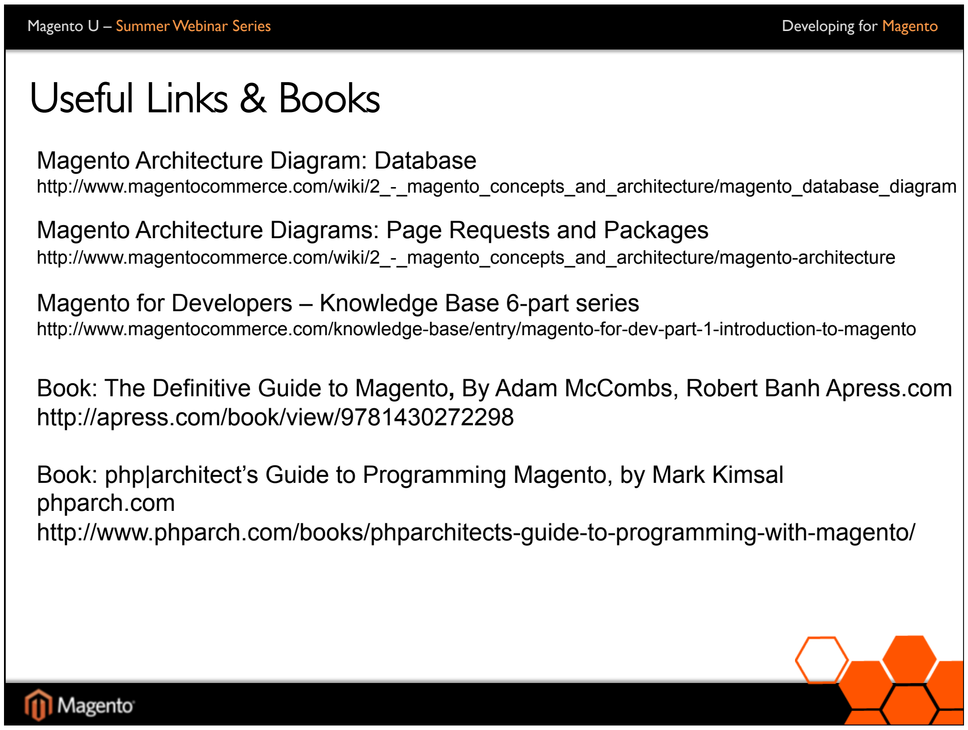
• Magento has a Model View Controller architecture, and can be fully customized through overrides.

• Modules can be created for overrides in multi-store environments.

• Magento core code should never be modified directly.

• Finding the correct code to modify involves understanding how to use Magento's code hints and how that correlates to Magento's file system and class structure.

Magento



Useful Links & Books

Magento Architecture Diagram: Database
http://www.magentocommerce.com/wiki/2_-_magento_concepts_and_architecture/magento_database_diagram

Magento Architecture Diagrams: Page Requests and Packages
http://www.magentocommerce.com/wiki/2_-_magento_concepts_and_architecture/magento-architecture

Magento for Developers – Knowledge Base 6-part series
http://www.magentocommerce.com/knowledge-base/entry/magento-for-dev-part-1-introduction-to-magento

Book: The Definitive Guide to Magento, By Adam McCombs, Robert Banh Apress.com
http://apress.com/book/view/9781430272298

Book: php|architect's Guide to Programming Magento, by Mark Kimsal
phparch.com
http://www.phparch.com/books/phparchitects-guide-to-programming-with-magento/

Magento

Questions / Comments?

Magento