

Magento for PHP MVC Developers

How I Learned to Stop Worrying and Love Magento

Who am I?

- Alan Storm
- <http://alanstorm.com>
- Got involved in the Internet/Web 1995
- Work in the Agency/Startup Space
- 10 years PHP experience
- The thing you can never say ...

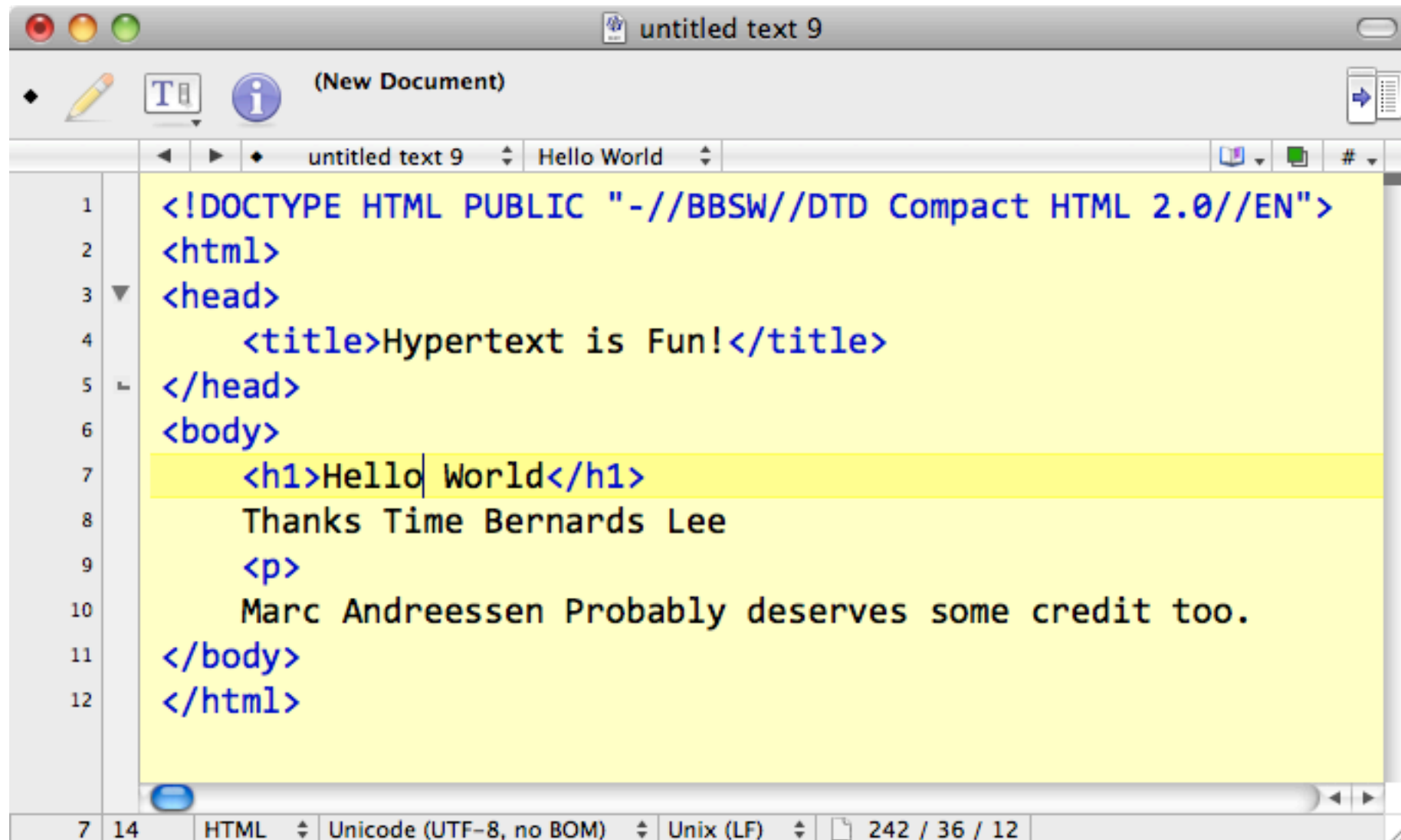
Not a Computer Scientist

- Many developers and programmers aren't
- You don't need to be a computer scientist to work with Magento
- Magento is similar to PHP Frameworks (Cake, CodeIgniter, Kohana), just deeper
- Magento and Kohana, side by side
- But first

How We Got Here

A Brief History of Web Development

HTML Page



The screenshot shows a text editor window titled "untitled text 9". The editor contains the following HTML code:

```
1 <!DOCTYPE HTML PUBLIC "-//BBSW//DTD Compact HTML 2.0//EN">
2 <html>
3 <head>
4     <title>Hypertext is Fun!</title>
5 </head>
6 <body>
7     <h1>Hello World</h1>
8     Thanks Time Bernards Lee
9     <p>
10     Marc Andreessen Probably deserves some credit too.
11 </body>
12 </html>
```

The status bar at the bottom indicates the file is 242 / 36 / 12 bytes, using HTML encoding, Unicode (UTF-8, no BOM), and Unix (LF) line endings.

HTML Page

- Very radical shift
- No programming needed, just markup
- No proprietary file format, just ASCII text

“Scripting” vs. “Real Programming”

- HTML included markup for form elements
- Stealth runtime, with client/server approach
- “Real Programmers” could do everything in C, C++, Java, lisp
- “Scripters” could piggy back on the web server for HTTP and produce HTML

Perl/CGI

- Perl was once the new hotness
- Excelled as string processing, and web applications are strings over HTTP
- CPAN meant a shared code library
- Non-complied meant distributed programs were shared in the open

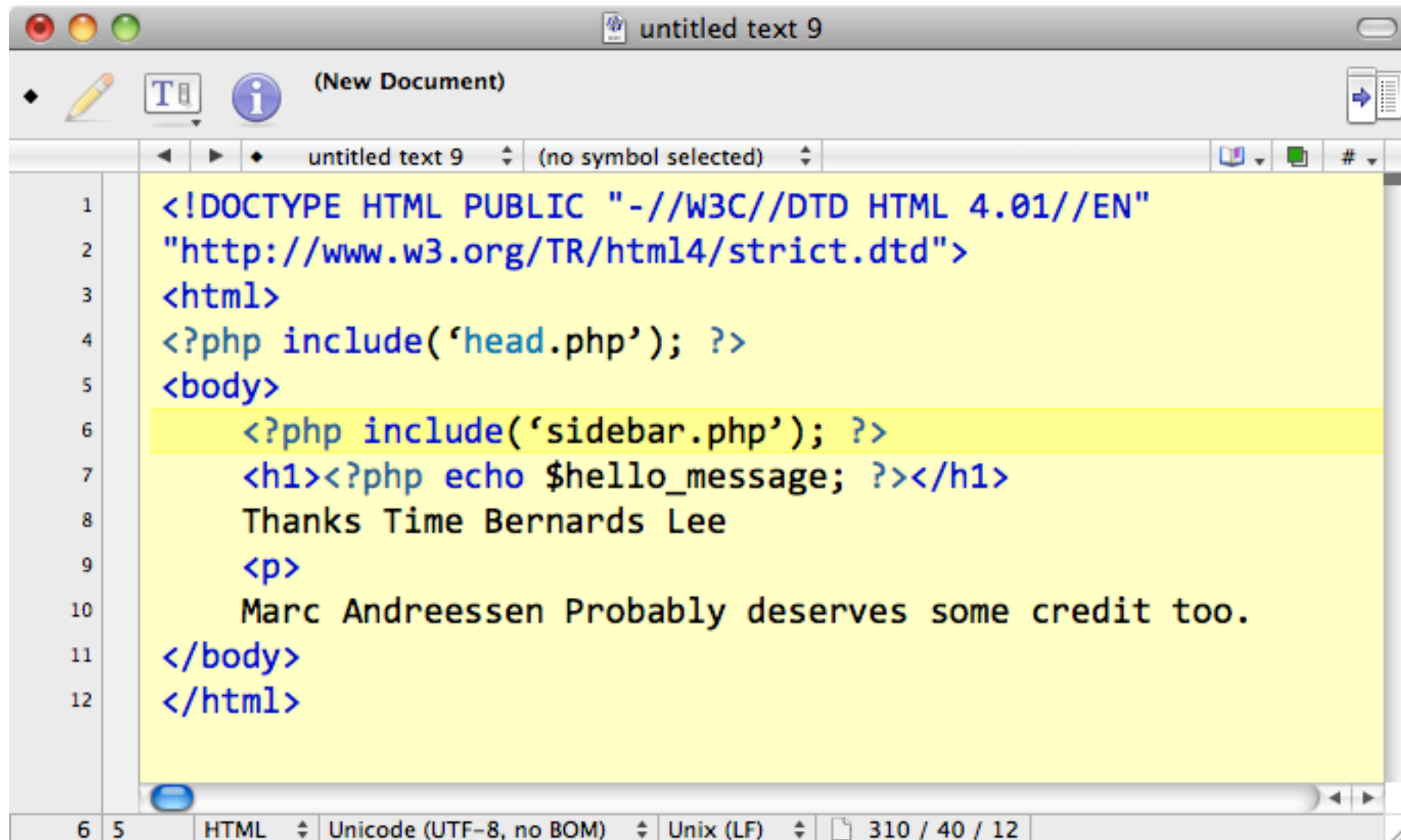
Perl/CGI

- CGI was slow, and FastCGI too new
- Perl didn't scale way before ruby didn't
- CPAN's good, but no standard installation for the growing shared hosting ecosystem

PHP 3/PHP 4

- Build important features directly into language, distributed by default
- Could import C code as extensions
- The build-up/tear-down approach of mod_php ideal for shared hosts
- Looks like Perl! Sort of.

PHP Page



```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2 "http://www.w3.org/TR/html4/strict.dtd">
3 <html>
4 <?php include('head.php'); ?>
5 <body>
6     <?php include('sidebar.php'); ?>
7     <h1><?php echo $hello_message; ?></h1>
8     Thanks Time Bernards Lee
9     <p>
10     Marc Andreessen Probably deserves some credit too.
11 </body>
12 </html>
```



```
1 <?php
2 $link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
3 if (!$link) {
4     die('Could not connect: ' . mysql_error());
5 }
6 ?>
7 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
8 "http://www.w3.org/TR/html4/strict.dtd">
9 <html>
10 <?php include('head.php'); ?>
11 <body>
12     <?php include('sidebar.php'); ?>
13     <h1><?php echo 'Hello World'; ?></h1>
14     <?php
15         $query = sprintf("SELECT firstname, lastname, address, age FROM friends WHERE firstname=
16             addslashes($firstname),
17             addslashes($lastname));
18
19         // Perform Query
20         $result = mysql_query($query);
21
22         while ($row = mysql_fetch_assoc($result)) {
23             echo $row['message'];
24             echo '<p>';
25         }
26     ?>
27 </body>
```

```
1 <?php
2 $link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
3 if (!$link) {
4     die('Could not connect: ' . mysql_error());
5 }
6 ?>
7 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
8 "http://www.w3.org/TR/html4/strict.dtd">
9 <html>
10 <?php include('head.php'); ?>
11 <body>
12     <?php
13     if(__FILE__ == 'main.php')
14     {
15         include('sidebar_main.php');
16     }
17     else if(__FILE__ == 'special.php')
18     {
19         include('special.php');
20     }
21     else if(__FILE__ == 'contact.php')
22     {
23         include('contact.php');
24     }
25     ?>
26     <h1><?php echo 'Hello World'; ?></h1>
27     <?php
```

1 1 HTML Unicode (UTF-8, no BOM) Unix (LF) 928 / 91 / 41

Rise of PHP Systems

- Created to combat PHP applications becoming too complex/messy
- Shopping Cart Applications
- CMS systems (Drupal, Mambo, Joomla)
- Programming Frameworks, many inspired by Ruby on Rails (CakePHP, Code Igniter, Symfony, etc.)

Magento is the Next Evolutionary Step

- Is a shopping cart application
- Uses a module based approach to load new code into the system, similar to Drupal
- Implements a configuration based MVC
- Most PHP Frameworks are convention based, Magento is configuration based

Controllers and Routing

Setting an Entry Point

MVC Routing

- Determines the main code entry point for any particular URL request
- Replaces PHP's one-page/one-url system
- All HTTP requests routed through a single index.php file
- Allows centralized library loading, and encourages separating business logic and layout code

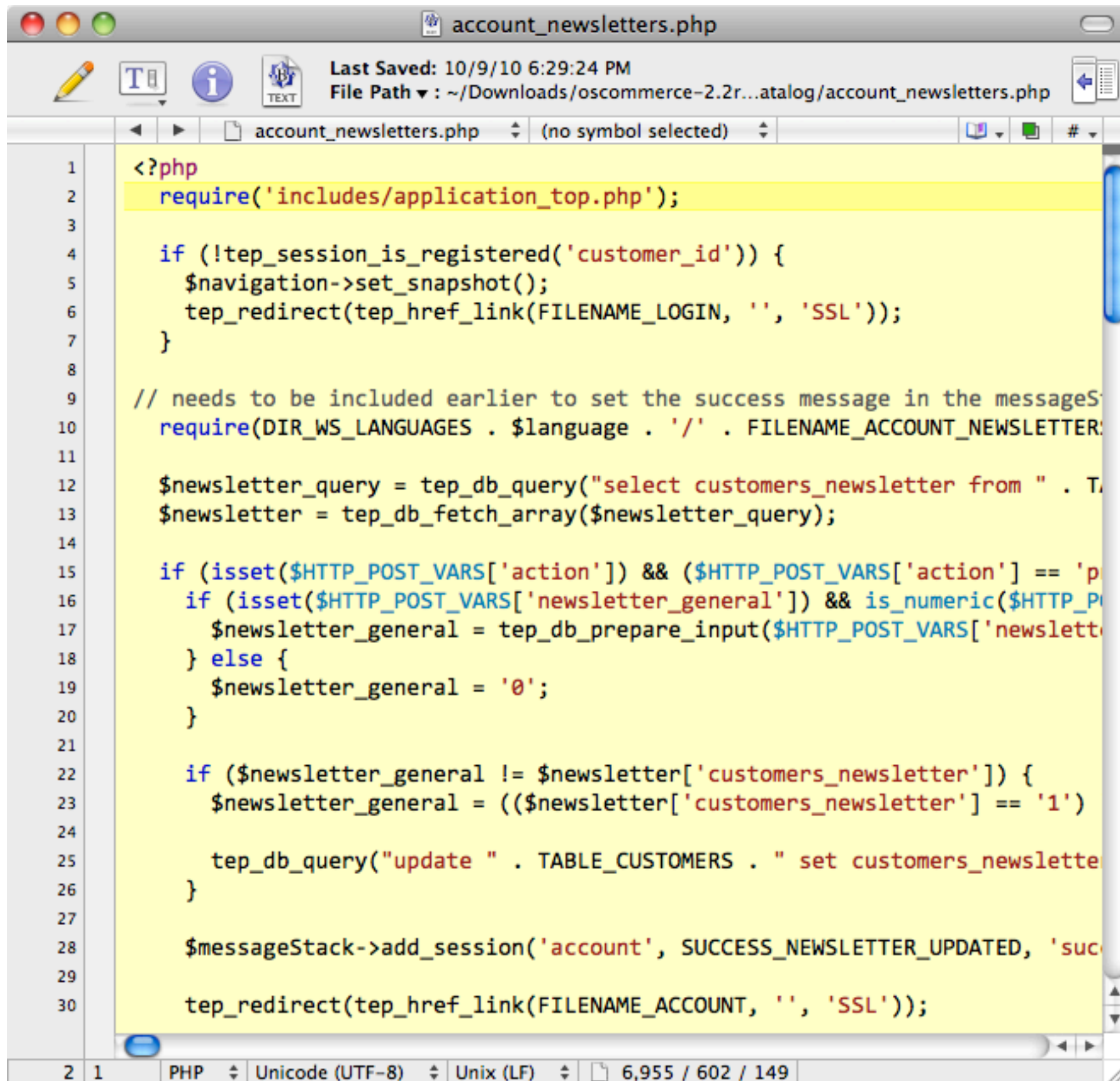
account_history.php

Last Saved: 10/9/10 6:28:48 PM
File Path: ~/Downloads/oscommerce-2.2rc2a/catalog/account_history.php

account_history.php (no symbol selected)

```
1 <?php
2     require('includes/application_top.php');
3
4     if (!tep_session_is_registered('customer_id')) {
5         $navigation->set_snapshot();
6         tep_redirect(tep_href_link(FILENAME_LOGIN, '', 'SSL'));
7     }
8
9     require(DIR_WS_LANGUAGES . $language . '/' . FILENAME_ACCOUNT_HISTORY);
10
11     $breadcrumb->add(NAVBAR_TITLE_1, tep_href_link(FILENAME_ACCOUNT, '', 'SSL'));
12     $breadcrumb->add(NAVBAR_TITLE_2, tep_href_link(FILENAME_ACCOUNT_HISTORY,
13 ?>
14 <!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
15 <html <?php echo HTML_PARAMS; ?>>
16 <head>
17 <meta http-equiv="Content-Type" content="text/html; charset=<?php echo CHAR
18 <title><?php echo TITLE; ?></title>
19 <base href="<?php echo (($request_type == 'SSL') ? HTTPS_SERVER : HTTP_SER
20 <link rel="stylesheet" type="text/css" href="stylesheet.css">
21 </head>
22 <body marginwidth="0" marginheight="0" topmargin="0" bottommargin="0" left
23 <!-- header //-->
24 <?php require(DIR_WS_INCLUDES . 'header.php'); ?>
25 <!-- header_eof //-->
26
27 <!-- body //-->
28 <table border="0" width="100%" cellpadding="3" cellspacing="3">
29 <tr>
30 <td width="<?php echo BOX_WIDTH; ?>" valign="top"><table border="0" wi
```

1 6 HTML Unicode (UTF-8) Unix (LF) 7,324 / 659 / 158



The image shows a text editor window titled "account_newsletters.php". The window has a menu bar with icons for editing, text, and information. The status bar at the top indicates "Last Saved: 10/9/10 6:29:24 PM" and "File Path: ~/Downloads/oscommerce-2.2r...atalog/account_newsletters.php". The editor shows a PHP script with line numbers 1 through 30 on the left. The code includes a require statement, session management, database queries, and a redirect. The bottom status bar shows "2 1 PHP Unicode (UTF-8) Unix (LF) 6,955 / 602 / 149".

```
1 <?php
2 require('includes/application_top.php');
3
4 if (!tep_session_is_registered('customer_id')) {
5     $navigation->set_snapshot();
6     tep_redirect(tep_href_link(FILENAME_LOGIN, '', 'SSL'));
7 }
8
9 // needs to be included earlier to set the success message in the messageStack
10 require(DIR_WS_LANGUAGES . $language . '/' . FILENAME_ACCOUNT_NEWSLETTER);
11
12 $newsletter_query = tep_db_query("select customers_newsletter from " . TABLE_CUSTOMERS);
13 $newsletter = tep_db_fetch_array($newsletter_query);
14
15 if (isset($HTTP_POST_VARS['action']) && ($HTTP_POST_VARS['action'] == 'update')) {
16     if (isset($HTTP_POST_VARS['newsletter_general']) && is_numeric($HTTP_POST_VARS['newsletter_general'])) {
17         $newsletter_general = tep_db_prepare_input($HTTP_POST_VARS['newsletter_general']);
18     } else {
19         $newsletter_general = '0';
20     }
21
22     if ($newsletter_general != $newsletter['customers_newsletter']) {
23         $newsletter_general = (($newsletter['customers_newsletter'] == '1') ? '0' : '1');
24
25         tep_db_query("update " . TABLE_CUSTOMERS . " set customers_newsletter = '" . $newsletter_general . "'");
26     }
27
28     $messageStack->add_session('account', SUCCESS_NEWSLETTER_UPDATED, 'success');
29
30     tep_redirect(tep_href_link(FILENAME_ACCOUNT, '', 'SSL'));
```

Kohana Routing

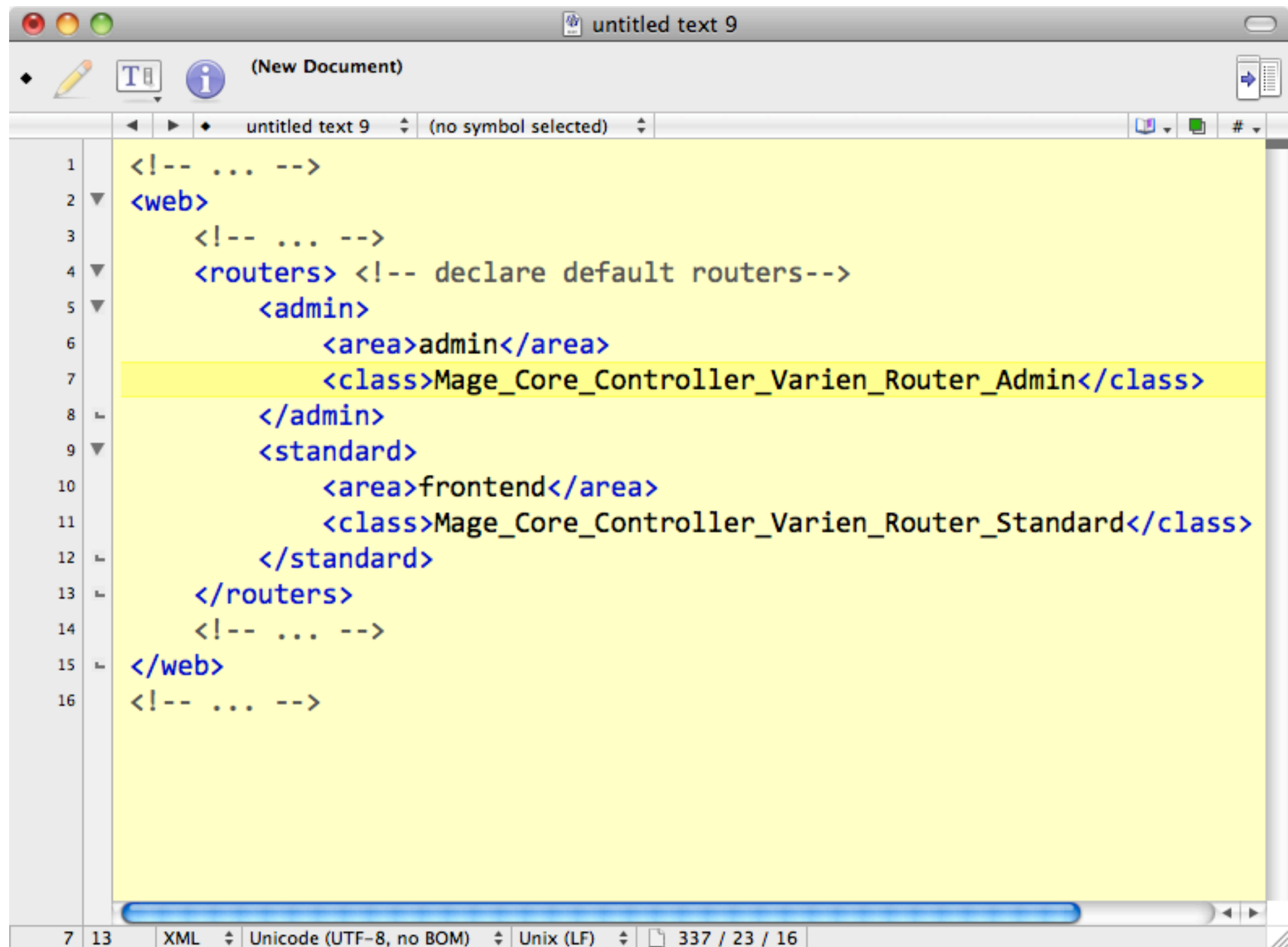
- Routing handled by Router_Core::setup
- URI: /controllerName/actionName
- URI: /welcome/index
- Welcome_Controller::index();
- File: /application/controllers/welcome.php

Magento Routing

- **URI: /frontName/controllerName/actionName**
- **URI: /catalogsearch/result/index**
- **No Standard Router Class!**

Magento Routing

- Magento Searches Global Config for routers
 - web/routers
- Default system ships with two Routers
 - Mage_Core_Controller_Varien_Router_Admin
 - Mage_Core_Controller_Varien_Router_Standard
- Configured in Mage_Core
 - /app/code/core/Mage/Core/etc/config.xml



The screenshot shows a text editor window with a title bar containing standard macOS window controls (red, yellow, green buttons) and the text 'untitled text 9'. Below the title bar is a toolbar with icons for a pencil, a text tool (T), and an information icon, followed by the text '(New Document)'. The main editing area has a light yellow background and contains XML code. A line number margin on the left shows numbers 1 through 16. The code is as follows:

```
1 <!-- ... -->
2 <web>
3     <!-- ... -->
4     <routers> <!-- declare default routers-->
5         <admin>
6             <area>admin</area>
7             <class>Mage_Core_Controller_Varien_Router_Admin</class>
8         </admin>
9         <standard>
10             <area>frontend</area>
11             <class>Mage_Core_Controller_Varien_Router_Standard</class>
12         </standard>
13     </routers>
14     <!-- ... -->
15 </web>
16 <!-- ... -->
```

At the bottom of the window is a status bar with the following information: '7 13' (likely line and column numbers), 'XML' (file encoding), 'Unicode (UTF-8, no BOM)' (character set), 'Unix (LF)' (line endings), and '337 / 23 / 16' (possibly file size or line/word/character counts).

Magento Routing

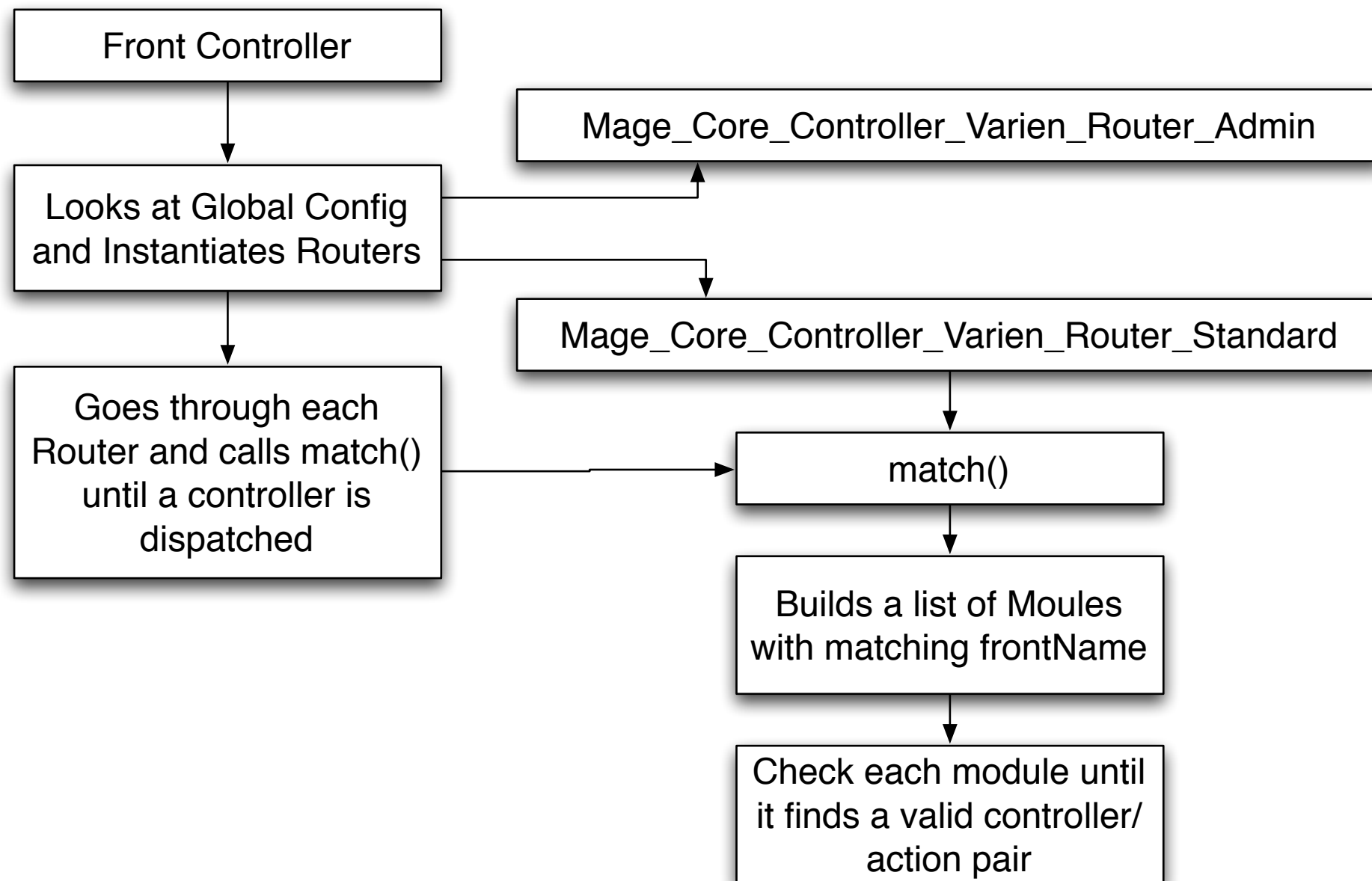
- The match method is called on each router
- The match method is responsible for
 - Creating a Controller class name
 - Instantiating the Controller
 - Creating an Action method
 - Calling the Action method

Magento Routing

- URI: /frontName/controllerName/actionName
- URI: /catalogsearch/result/index
- Default routers will
 - Identify any module with a configured `<frontName>catalogsearch</frontName>`
 - Check module's controllers folder for a `ResultController.php`

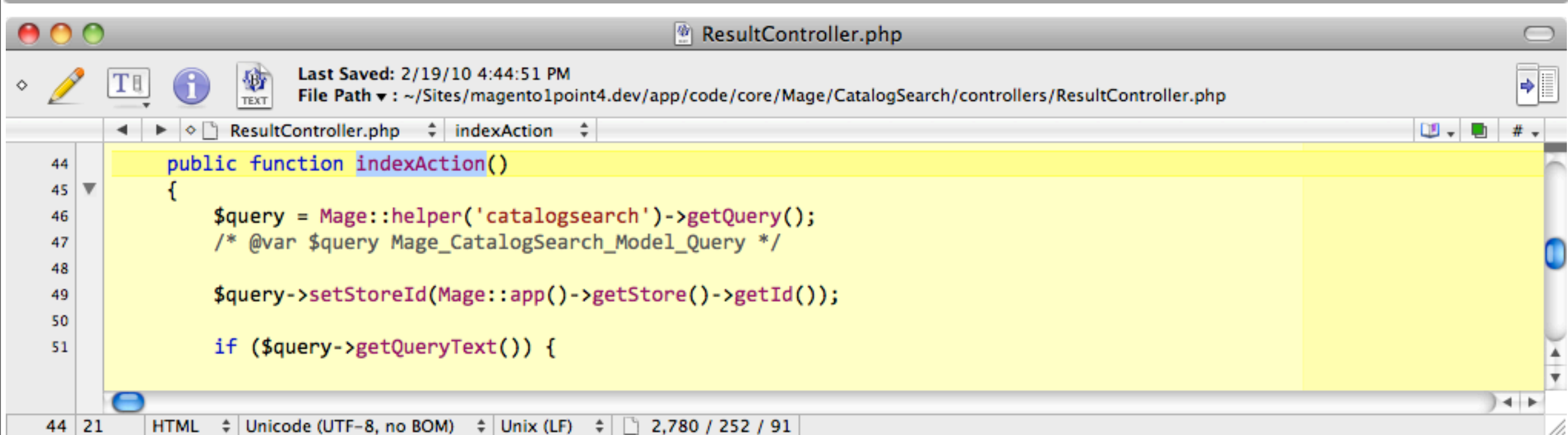
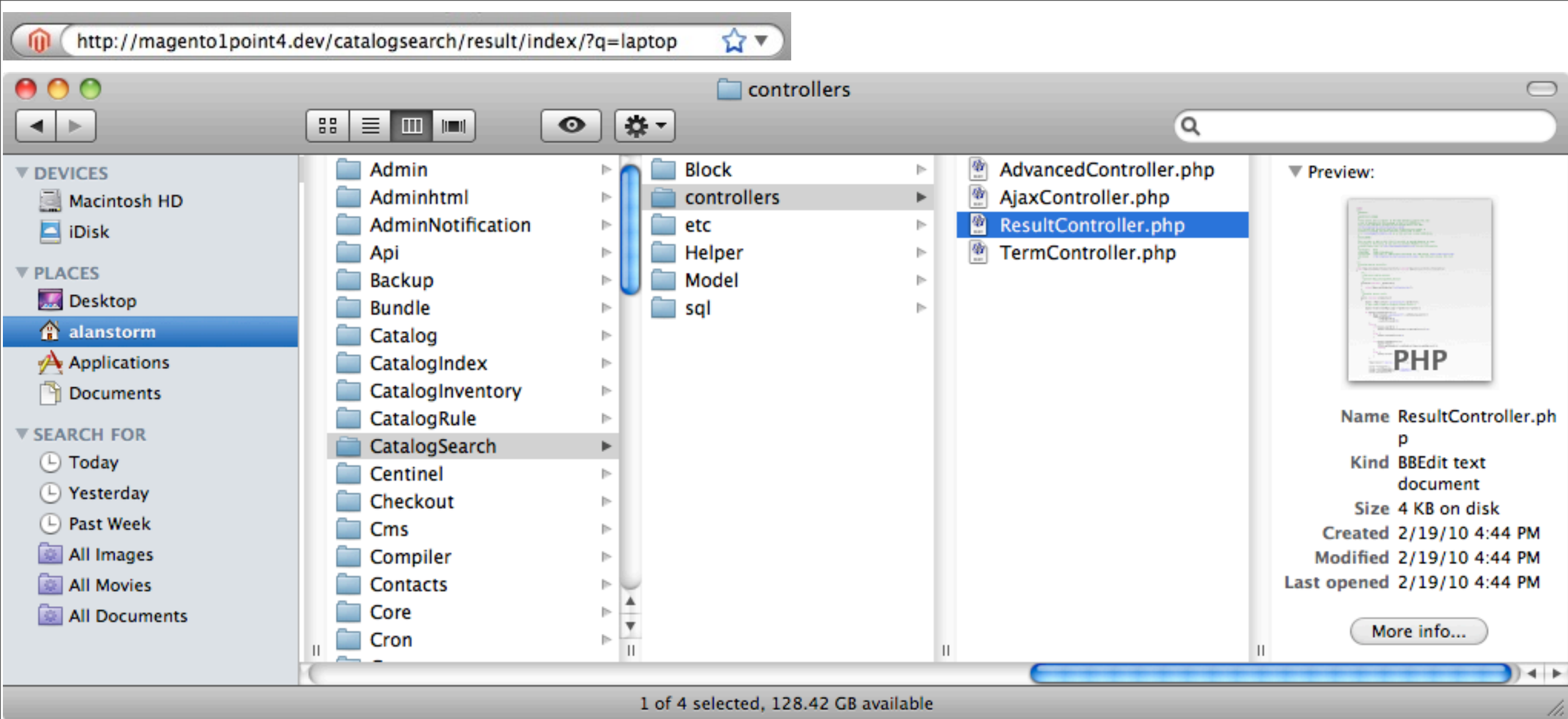
Magento Routing

- /frontName/controllerName/actionName
- /catalogsearch/result/index
- If a Controller is found, Router checks if for an action method
 - Mage_CatalogSearch_ResultController::indexAction
- If action is found, it's called, and the module search/match stops.



Magento Routing

- Configuration base system, but a de-facto convention exists
- `/catalogsearch/result/index`
- translates to
 - Module: *Mage_CatalogSearch*
 - Controller: *ResultController*
 - Action: *indexAction*



Custom Routing

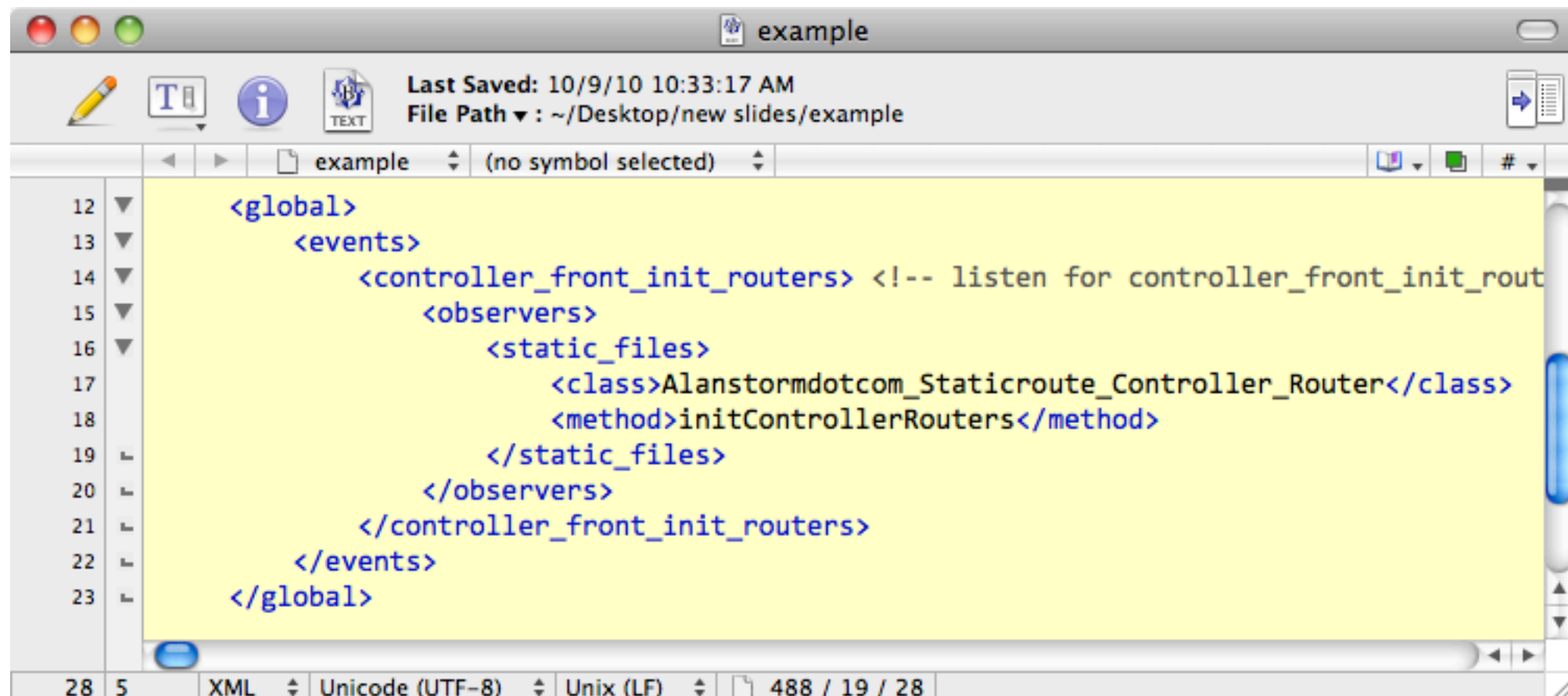
- Could add directly to your module's `<outers /> config`
- Also possible to add additional Routers through event system
 - `controller_front_init_routers` event

Practical Example

- Packaging front-end files within a module directories
 - `app/code/local/Alanstormdotcom/Staticroute/static-frontend/js/example.js`
 - `app/code/local/Alanstormdotcom/Staticroute/static-frontend/css/example.css`
- Create a custom Router class
- Add match rules to intercept URLs in the form
 - `http://example.com/static-frontend/css/example.css`
- Connect URL for full code
 - `http://www.magentocommerce.com/magento-connect/developer/alanstorm`

Adding A Router

- Setting up an event listener
- Event handling class is also your Router



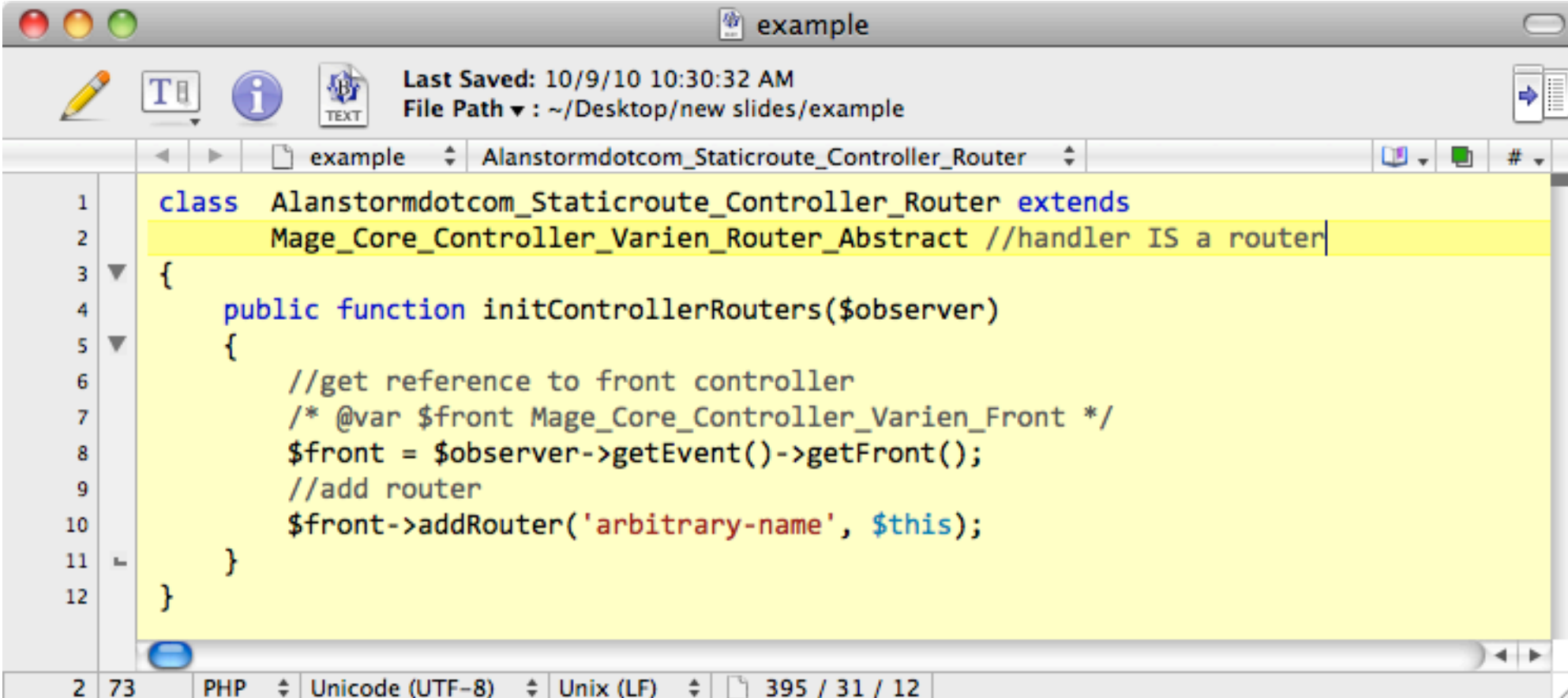
The screenshot shows a text editor window titled "example" with a toolbar and status bar. The main text area contains XML code for configuring a router. The code is as follows:

```
12 <global>
13   <events>
14     <controller_front_init_routers> <!-- listen for controller_front_init_rout
15       <observers>
16         <static_files>
17           <class>Alanstormdotcom_Staticroute_Controller_Router</class>
18           <method>initControllerRouters</method>
19         </static_files>
20       </observers>
21     </controller_front_init_routers>
22   </events>
23 </global>
```

The status bar at the bottom indicates the file is 488 / 19 / 28 bytes, using XML encoding, Unicode (UTF-8) characters, and Unix (LF) line endings.

Implement Router

- Extend base Router Class
 - Mage_Core_Controller_Varien_Router_Abstract
- Implement Event Handling Method

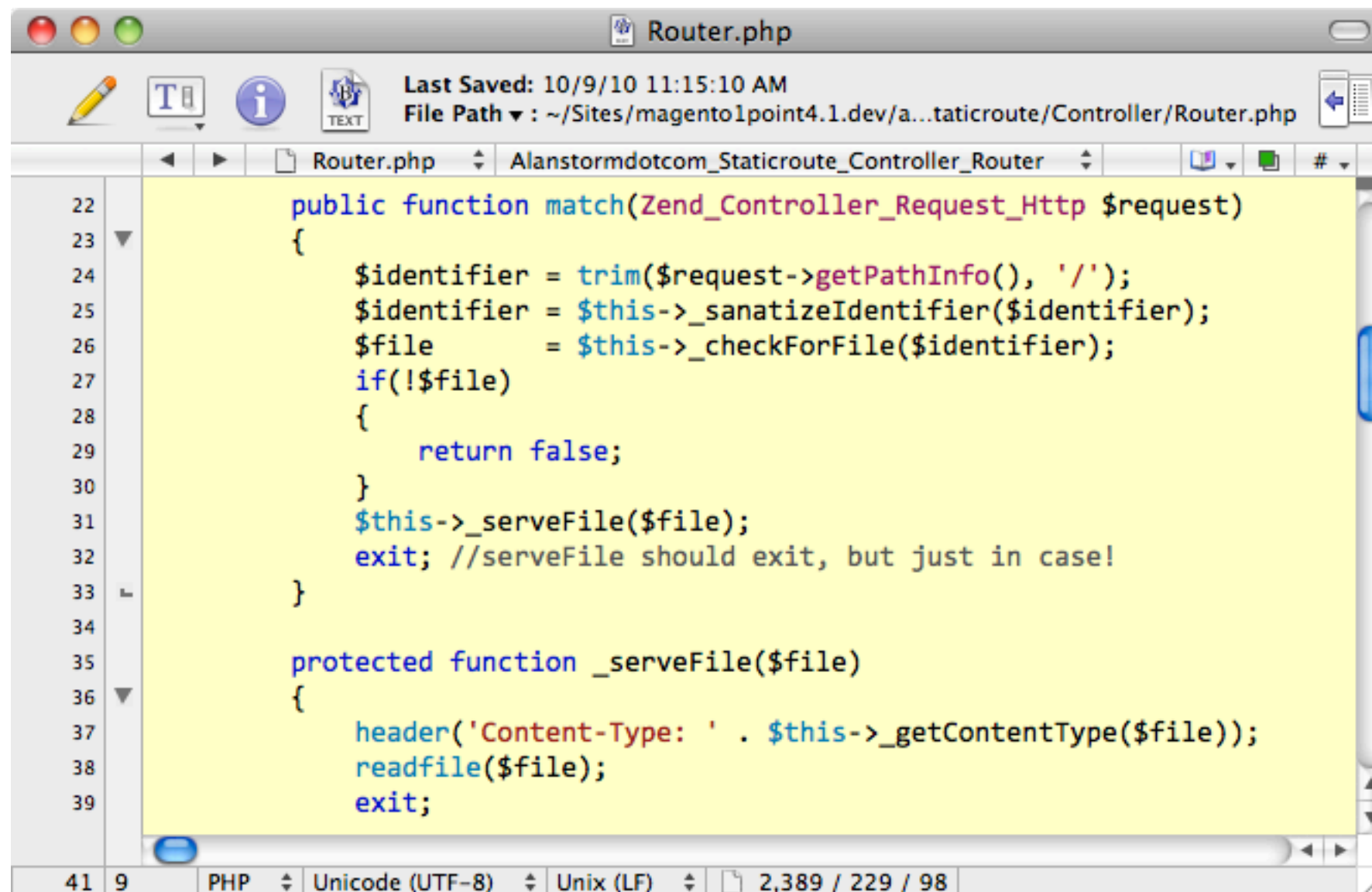


The screenshot shows a code editor window titled 'example'. The editor displays PHP code for a class named 'Alanstormdotcom_Staticroute_Controller_Router' which extends 'Mage_Core_Controller_Varien_Router_Abstract'. The code includes a comment '//handler IS a router' and a public function 'initControllerRouters(\$observer)' that sets up a front controller and adds the current router to it. The status bar at the bottom indicates the file is 395 / 31 / 12 bytes, using PHP, Unicode (UTF-8), and Unix (LF) line endings.

```
1 class Alanstormdotcom_Staticroute_Controller_Router extends
2     Mage_Core_Controller_Varien_Router_Abstract //handler IS a router
3 {
4     public function initControllerRouters($observer)
5     {
6         //get reference to front controller
7         /* @var $front Mage_Core_Controller_Varien_Front */
8         $front = $observer->getEvent()->getFront();
9         //add router
10        $front->addRouter('arbitrary-name', $this);
11    }
12 }
```

Implement match

- Use \$identifier to check for file in match method
 - \$identifier = '/static-frontend/css/example.css'



The screenshot shows a code editor window titled 'Router.php'. The editor displays the following PHP code:

```
22     public function match(Zend_Controller_Request_Http $request)
23     {
24         $identifier = trim($request->getPathInfo(), '/');
25         $identifier = $this->_sanitizeIdentifier($identifier);
26         $file       = $this->_checkForFile($identifier);
27         if(!$file)
28         {
29             return false;
30         }
31         $this->_serveFile($file);
32         exit; //serveFile should exit, but just in case!
33     }
34
35     protected function _serveFile($file)
36     {
37         header('Content-Type: ' . $this->_getContentType($file));
38         readfile($file);
39         exit;
```

The code editor interface includes a toolbar with icons for editing, a status bar at the bottom showing 'PHP', 'Unicode (UTF-8)', 'Unix (LF)', and file statistics '2,389 / 229 / 98'.

Models

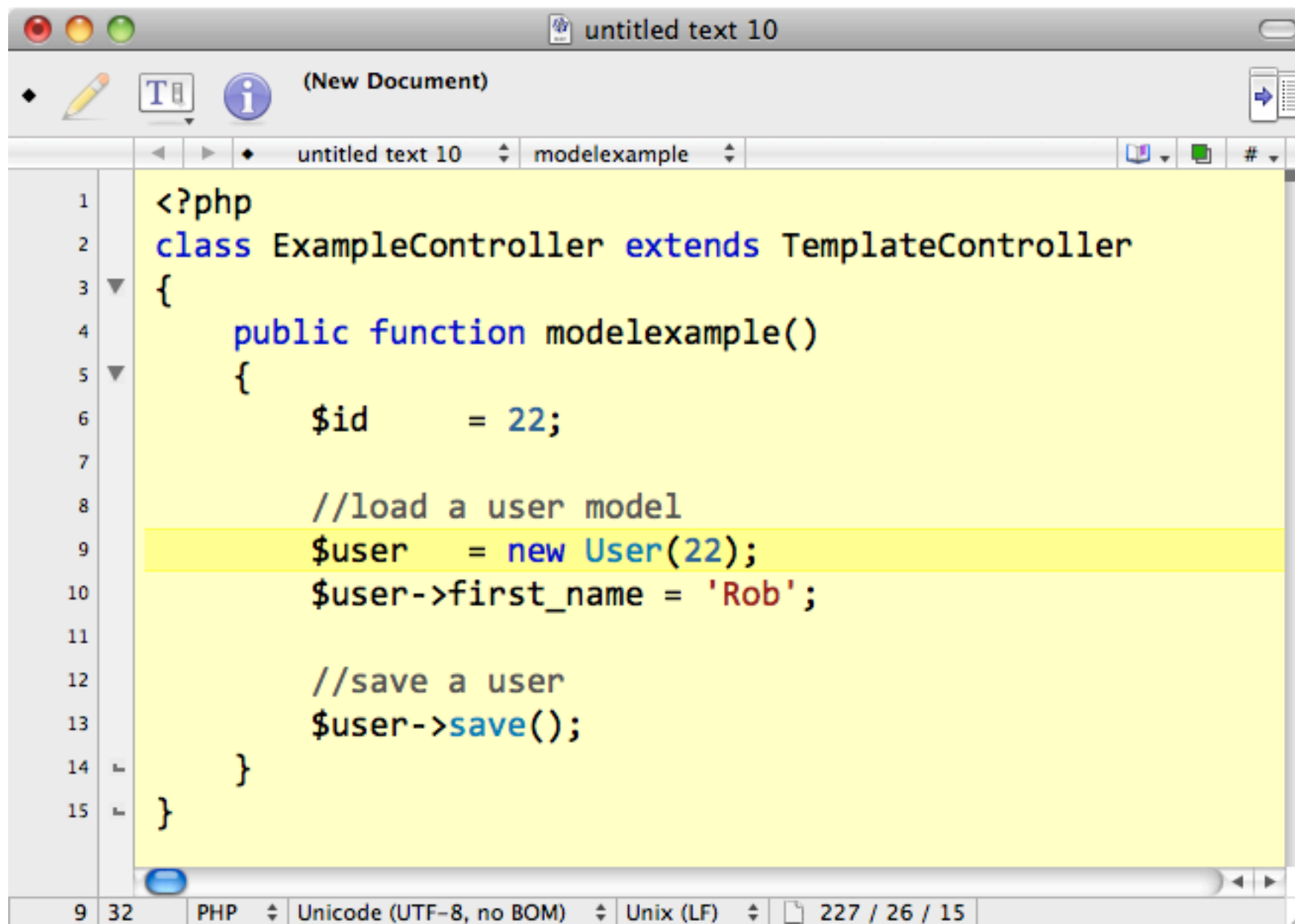
Getting out of the SQL Writing Business

MVC Models

- MVC Model are classes that represent the application's data
- Use to organize and/or automate data queries (SQL) for cleaner code
- Also provide centralized location for query optimizations
- Encapsulate business logic in methods

Kohana Models

- Extend from Base ORM class
- Mostly follows Active Record Pattern
 - Create, Read, Update, Delete
- Simple Class Instantiation



```
<?php
class ExampleController extends TemplateController
{
    public function modelexample()
    {
        $id      = 22;

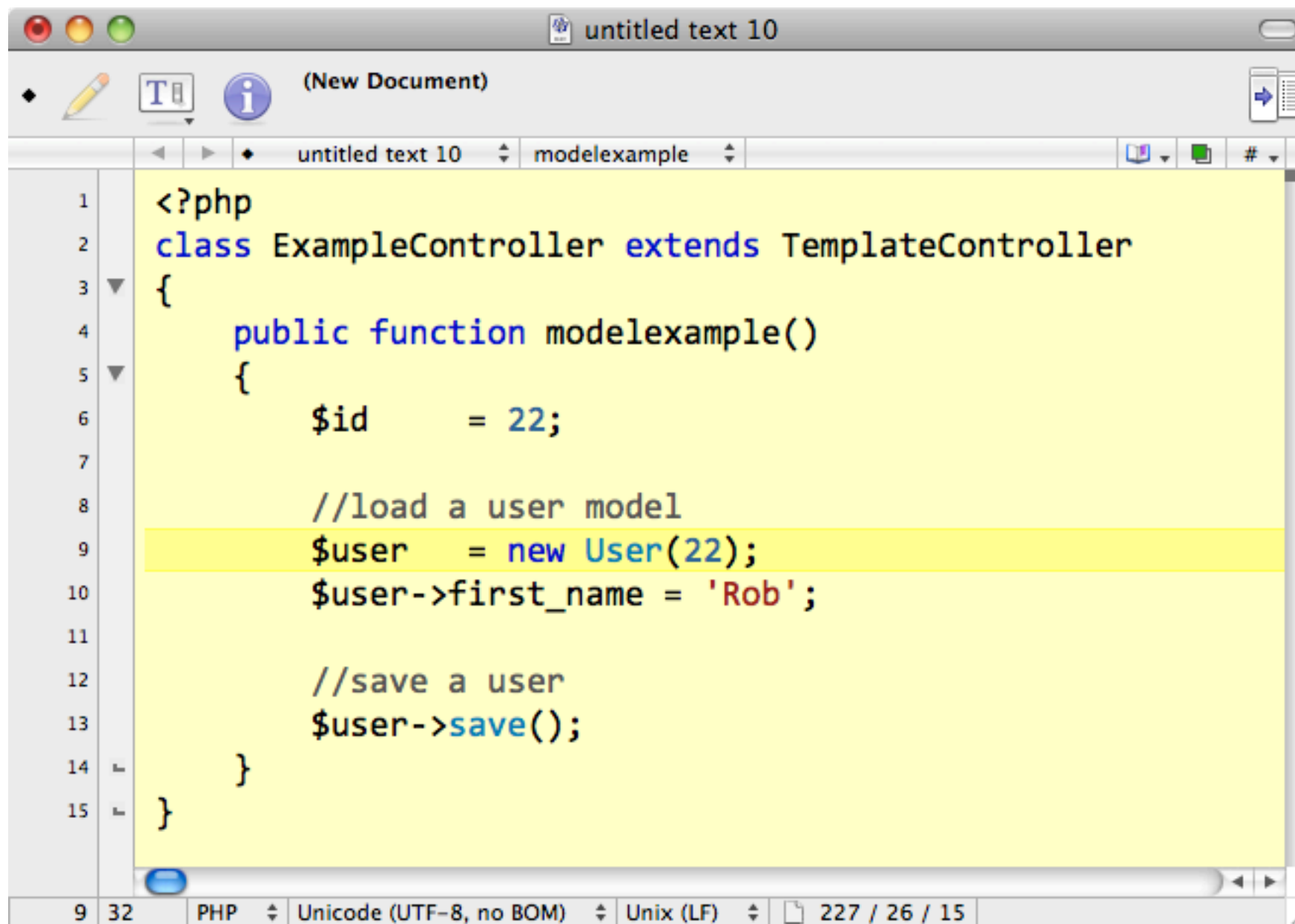
        //load a user model
        $user     = new User(22);
        $user->first_name = 'Rob';

        //save a user
        $user->save();
    }
}
```

9 32 PHP Unicode (UTF-8, no BOM) Unix (LF) 227 / 26 / 15

Magento Models

- Many Models implement an Entity Attribute Value store
- Simpler “basic” models offers one table/one class Active Record like functionality
- Implement a Mapper pattern, making Model independent of data store
- Client Programmer doesn't need to know these details to use.



```
<?php
class ExampleController extends TemplateController
{
    public function modelexample()
    {
        $id      = 22;

        //load a user model
        $user     = new User(22);
        $user->first_name = 'Rob';

        //save a user
        $user->save();
    }
}
```

9 32 PHP Unicode (UTF-8, no BOM) Unix (LF) 227 / 26 / 15

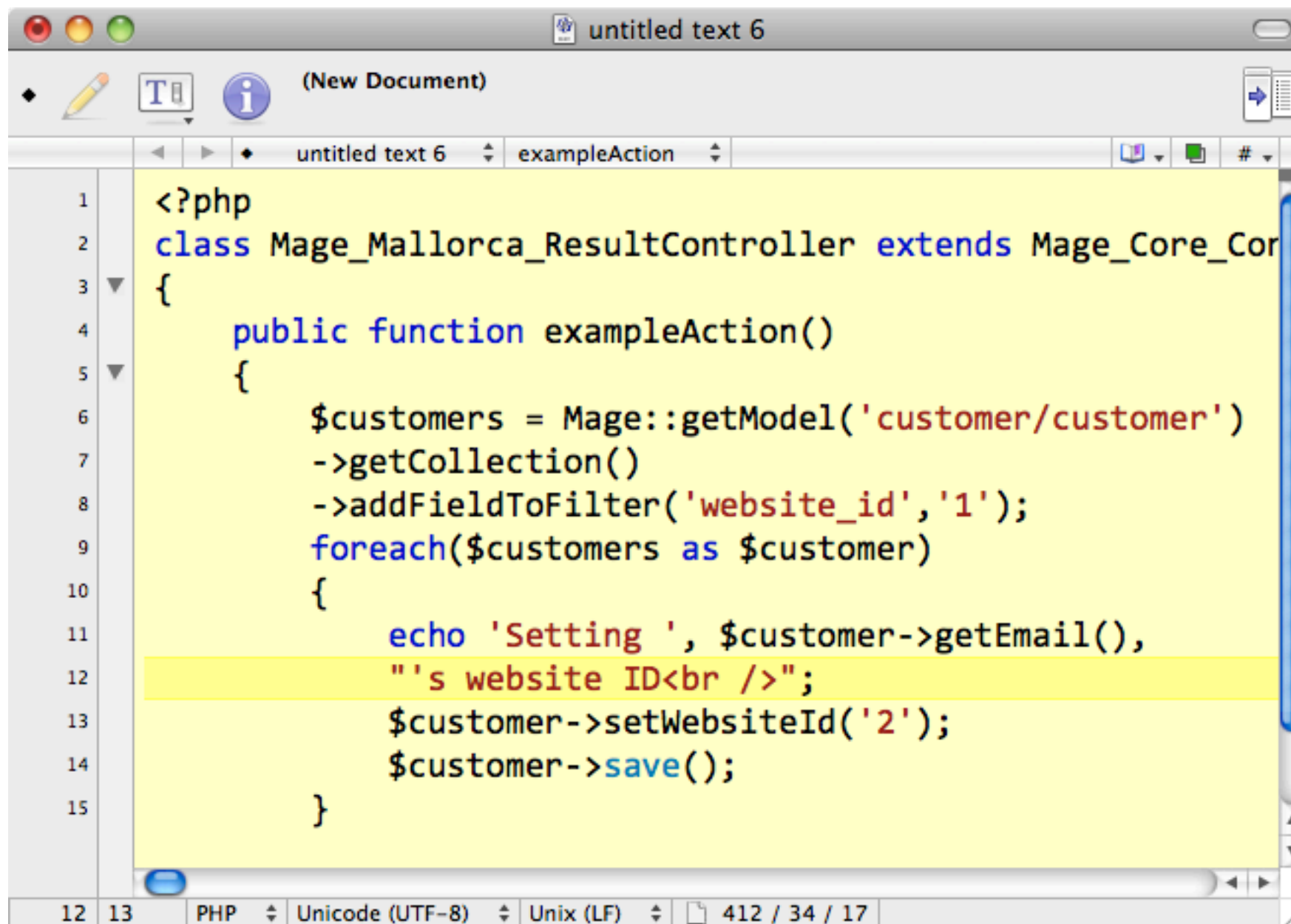

```
<?php
class Mage_Mallorca_ResultController extends Mage_Core_Controller_Front_Action
{
    public function exampleAction()
    {
        $id      = 22;

        //load a user model
        $customer = Mage::getModel('customer/customer')
->load($id);
        $customer->setFirstName('Rob');

        //save a user
        $customer->save();
    }
}
```

Magento Models

- Magento Replaces direct instantiation with an indirect static method call
 - `$user = new User();`
 - `$customer = Mage::getModel('customer/customer');`
- Every model has a Collection object which is used when fetching for multiple models. Kohana and other frameworks, return PHP arrays()



```
<?php
class Mage_Mallorca_ResultController extends Mage_Core_Controller_Front_Action
{
    public function exampleAction()
    {
        $customers = Mage::getModel('customer/customer')
            ->getCollection()
            ->addFieldToFilter('website_id','1');
        foreach($customers as $customer)
        {
            echo 'Setting ', $customer->getEmail(),
                "'s website ID<br />";
            $customer->setWebsiteId('2');
            $customer->save();
        }
    }
}
```

Magento Collections

- Collections are objects that act like arrays
- Ancestors implement Standard PHP Library Interfaces: IteratorAggregate and Countable
- Inherit from class `Varien_Data_Collection`
- Allows you to encapsulate business logic with custom methods on collections

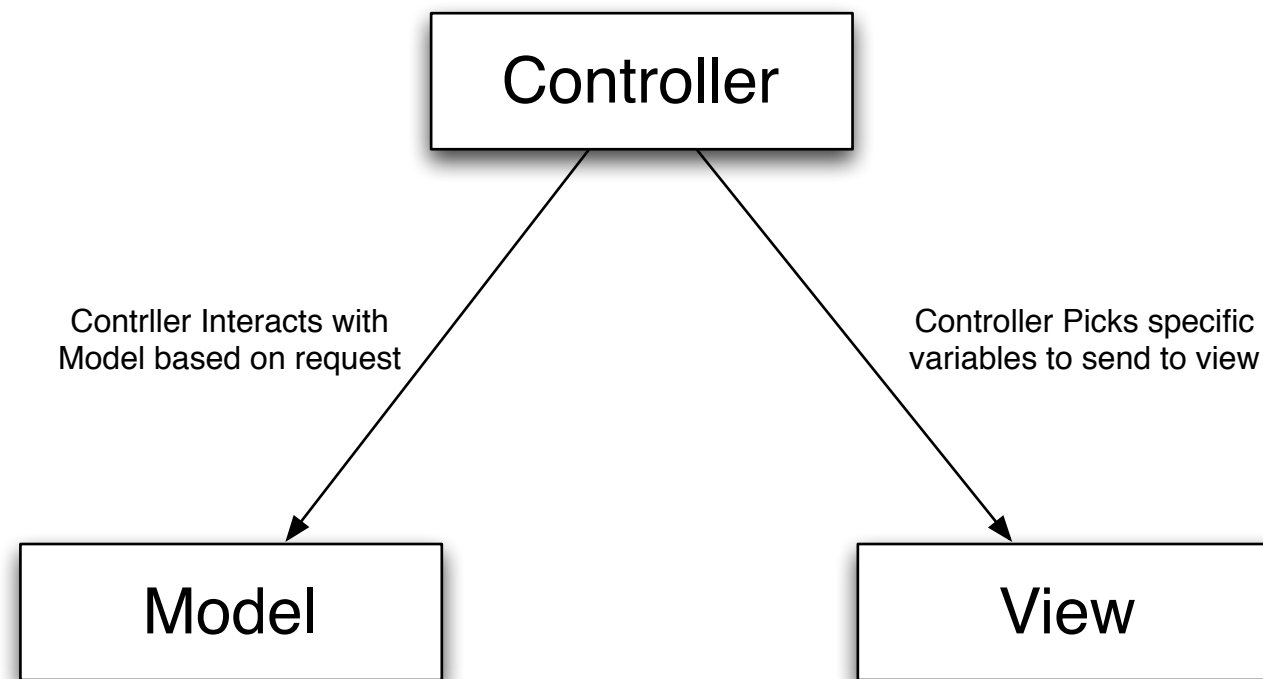
Views

It all comes back to HTML

MVC Views

- Views create the response output
- Typically HTML
- Could be JSON, XML, etc.
- Separates template logic from application

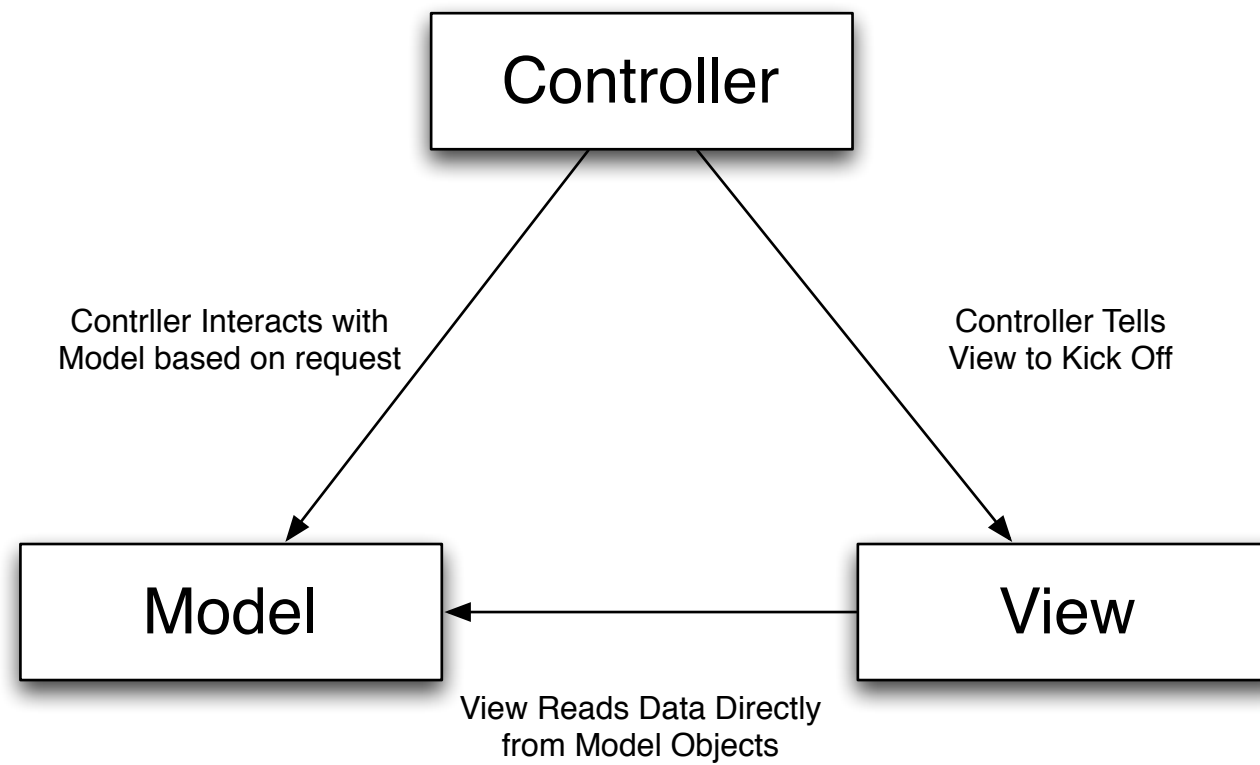
Kohana View

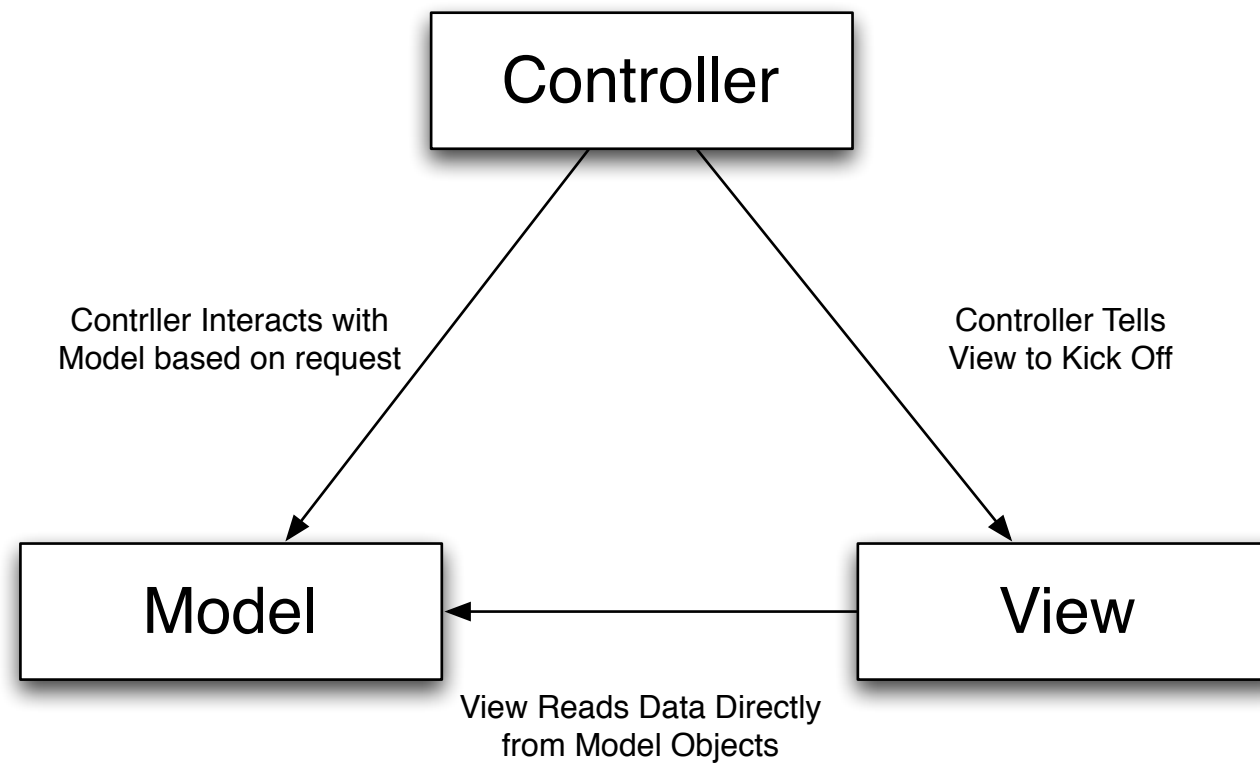


Kohana View

- Traditional PHP MVC View Pattern
- Views are PHP mixed with HTML
- Often called “Dumb View”, since the View doesn’t know anything about the Application

Magento View

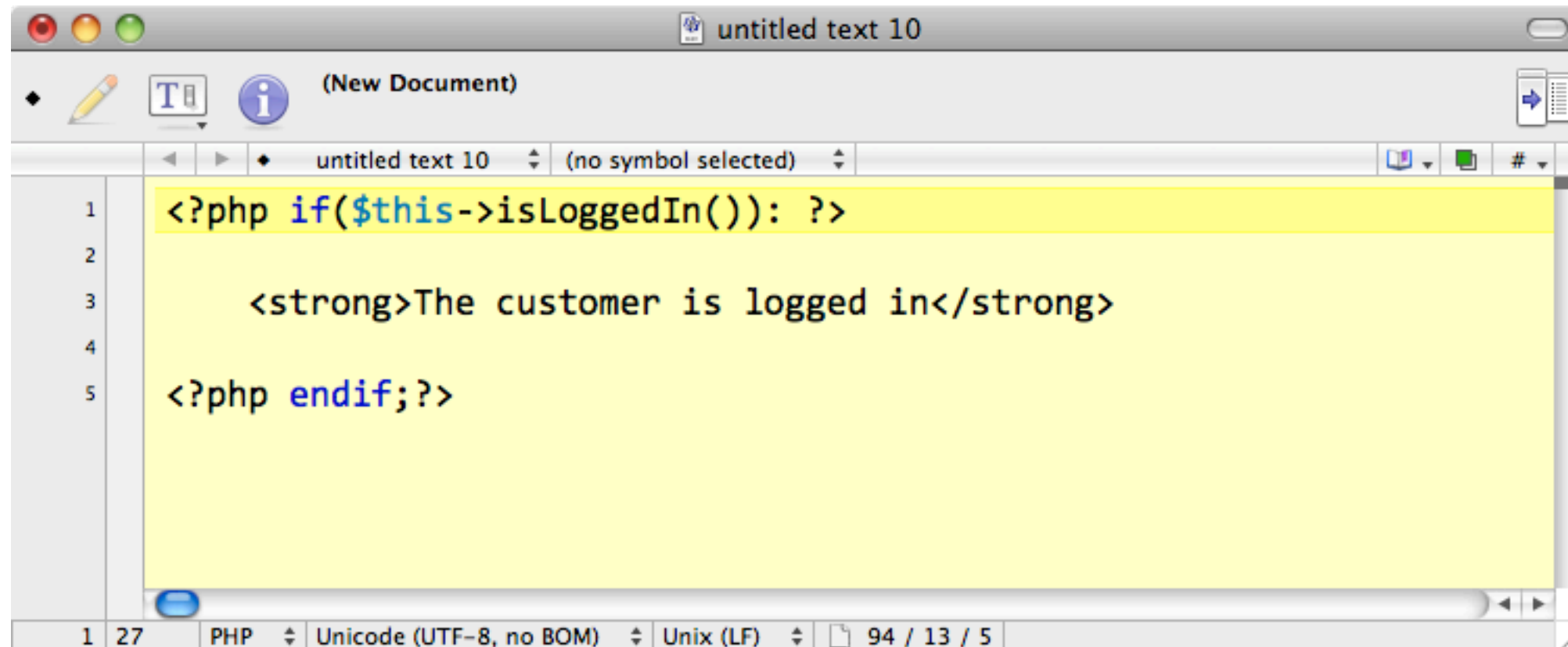




Magento Blocks

- The simplest block is just a PHP class with a `toHtml()` method, which returns a string
- Many Blocks render via a phtml template
- From within a phtml template, `$this` is a reference to the containing Block
- The phtml templates contain HTML and PHP code for rendering logic
- Block methods communicate with Models

Template for Block



A screenshot of a text editor window titled "untitled text 10". The editor shows a PHP template for a block. The code is as follows:

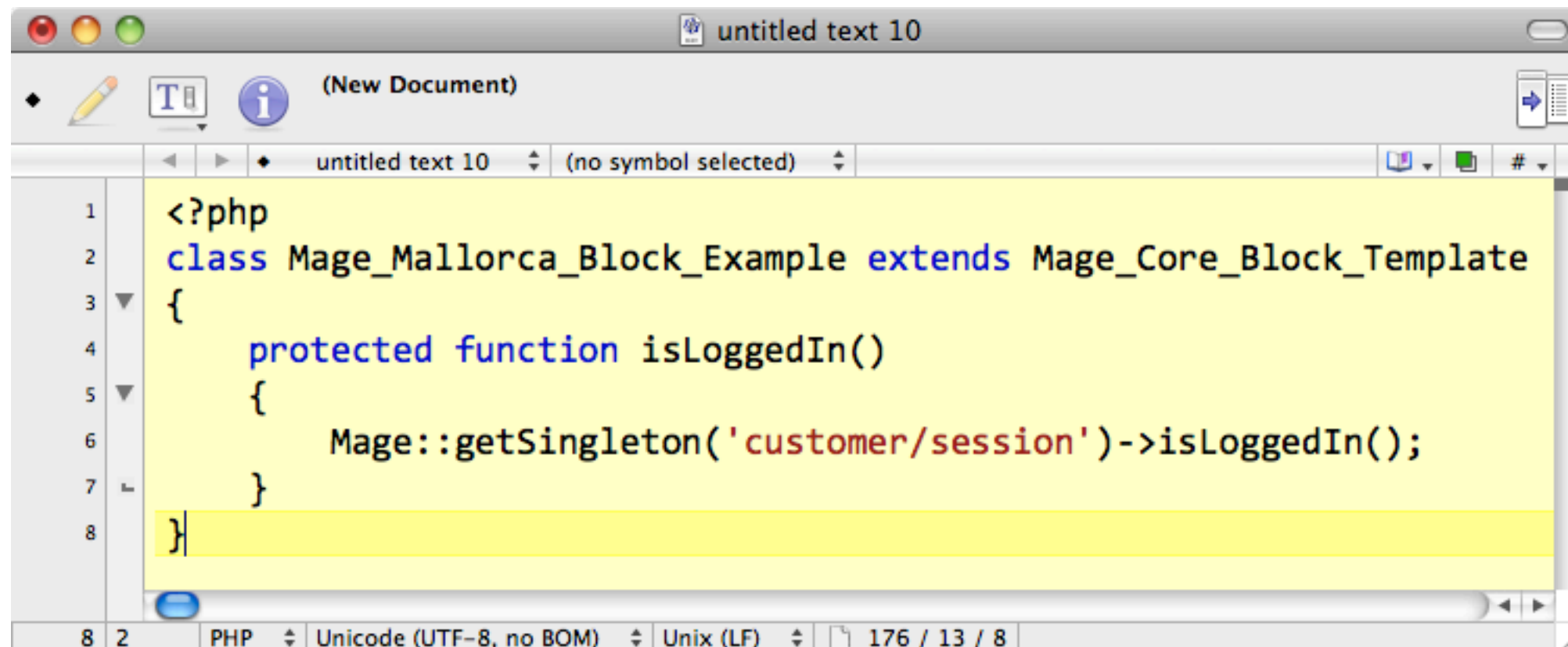
```
<?php if($this->isLoggedIn()): ?>

    <strong>The customer is logged in</strong>

<?php endif;?>
```

The status bar at the bottom indicates the file is in PHP mode, using Unicode (UTF-8, no BOM) encoding, with Unix (LF) line endings. The cursor is at line 1, column 27.

Block Class



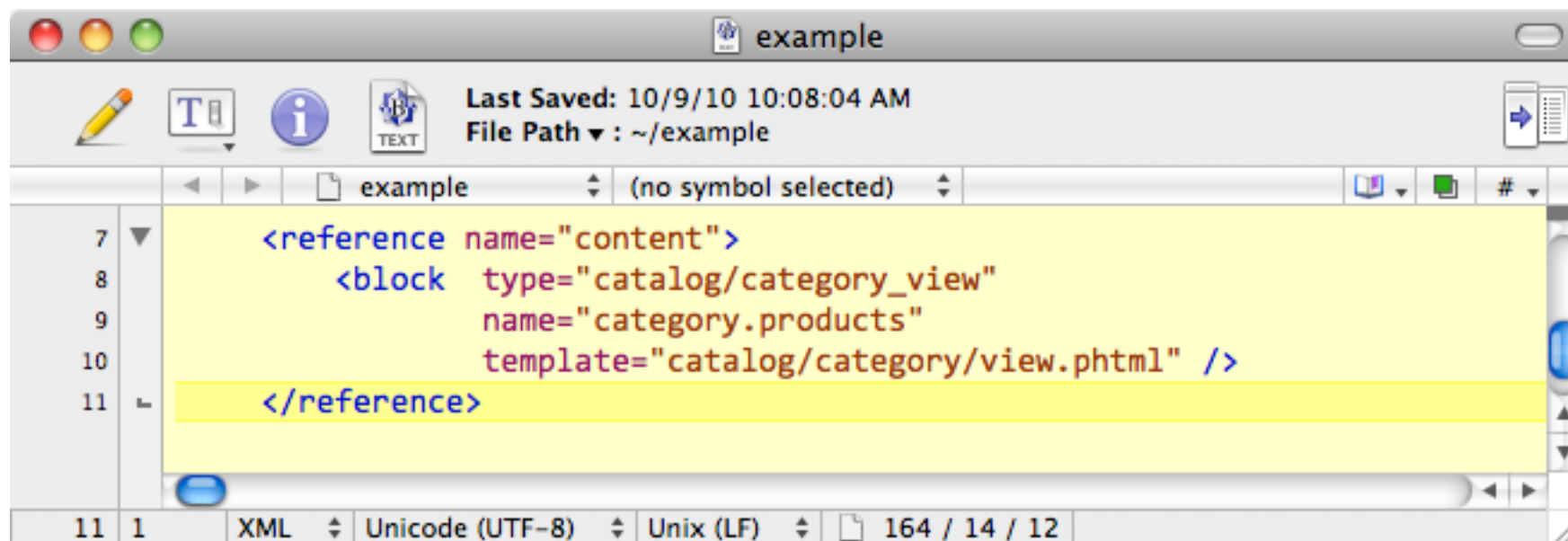
A screenshot of a text editor window titled "untitled text 10". The editor shows a PHP class definition for a block. The code is as follows:

```
<?php
class Mage_Mallorca_Block_Example extends Mage_Core_Block_Template
{
    protected function isLoggedIn()
    {
        Mage::getSingleton('customer/session')->isLoggedIn();
    }
}
```

The status bar at the bottom indicates the file is in PHP mode, using Unicode (UTF-8, no BOM) encoding, with Unix (LF) line endings. The cursor is at line 8, column 2.

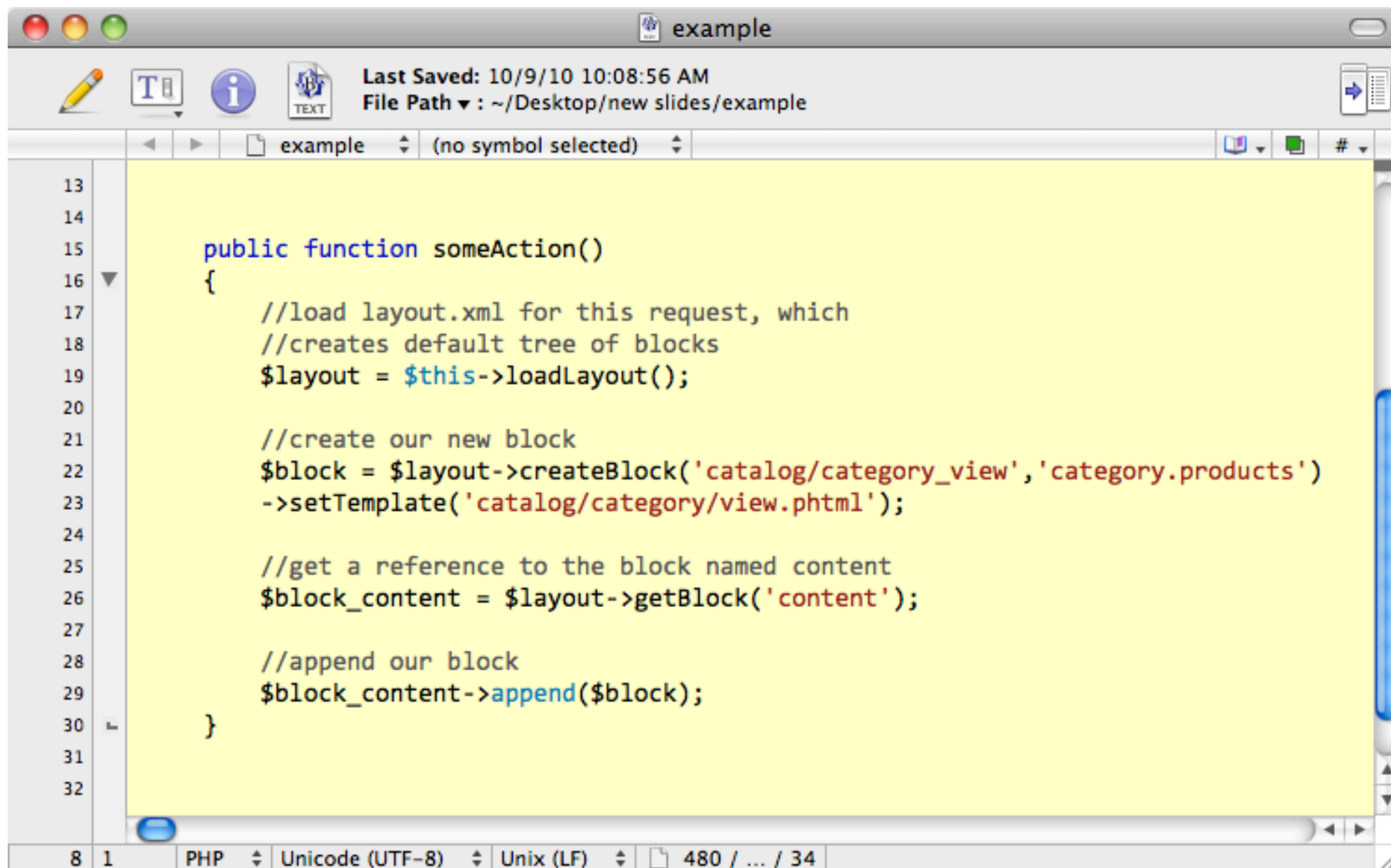
Magento Layout

- A single Layout object contains all the Block objects for a particular request
- Layout XML files allows users to control which Blocks are available for any request
 - Layout XML system allows designers and interactive developers to fully control page structure without editing Controllers or Models
- However, Blocks can be added outside the Layout XML system using PHP



This screenshot shows an XML editor window titled "example". The status bar at the top indicates "Last Saved: 10/9/10 10:08:04 AM" and "File Path: ~/example". The editor displays XML code with line numbers 7 through 11 on the left. The code defines a reference named "content" which contains a block of type "catalog/category_view" with the name "category.products" and template "catalog/category/view.phtml". The status bar at the bottom shows "11 1 XML Unicode (UTF-8) Unix (LF) 164 / 14 / 12".

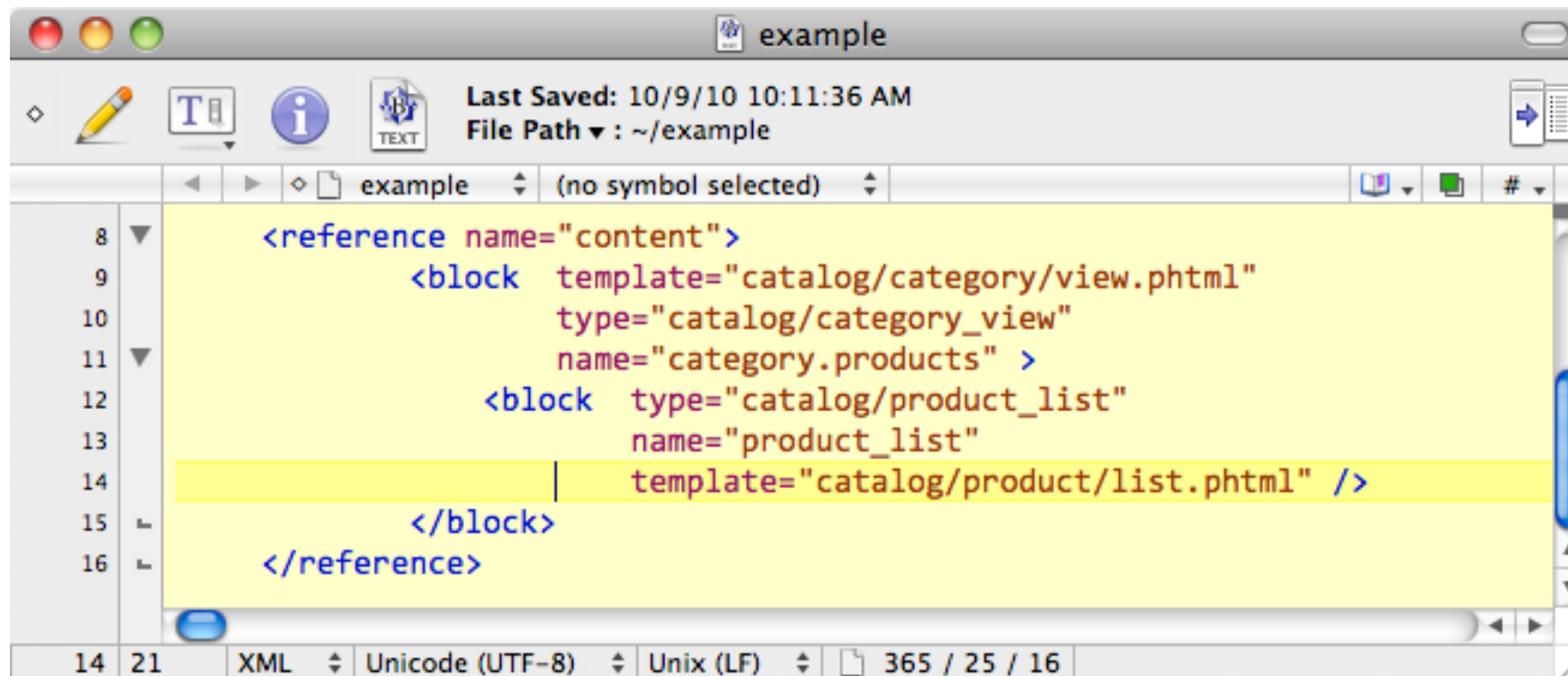
```
7 <reference name="content">
8   <block type="catalog/category_view"
9         name="category.products"
10        template="catalog/category/view.phtml" />
11 </reference>
```



This screenshot shows a PHP editor window titled "example". The status bar at the top indicates "Last Saved: 10/9/10 10:08:56 AM" and "File Path: ~/Desktop/new slides/example". The editor displays PHP code with line numbers 13 through 32 on the left. The code defines a public function "someAction()" which loads a layout, creates a block, gets a reference to the "content" block, and appends the new block to it. The status bar at the bottom shows "8 1 PHP Unicode (UTF-8) Unix (LF) 480 / ... / 34".

```
13
14
15 public function someAction()
16 {
17     //load layout.xml for this request, which
18     //creates default tree of blocks
19     $layout = $this->loadLayout();
20
21     //create our new block
22     $block = $layout->createBlock('catalog/category_view', 'category.products')
23     ->setTemplate('catalog/category/view.phtml');
24
25     //get a reference to the block named content
26     $block_content = $layout->getBlock('content');
27
28     //append our block
29     $block_content->append($block);
30 }
31
32
```

Child Blocks



The screenshot shows a text editor window titled 'example'. The menu bar includes 'Last Saved: 10/9/10 10:11:36 AM' and 'File Path: ~/example'. The toolbar shows icons for undo, redo, and other editing functions. The main text area contains the following XML code:

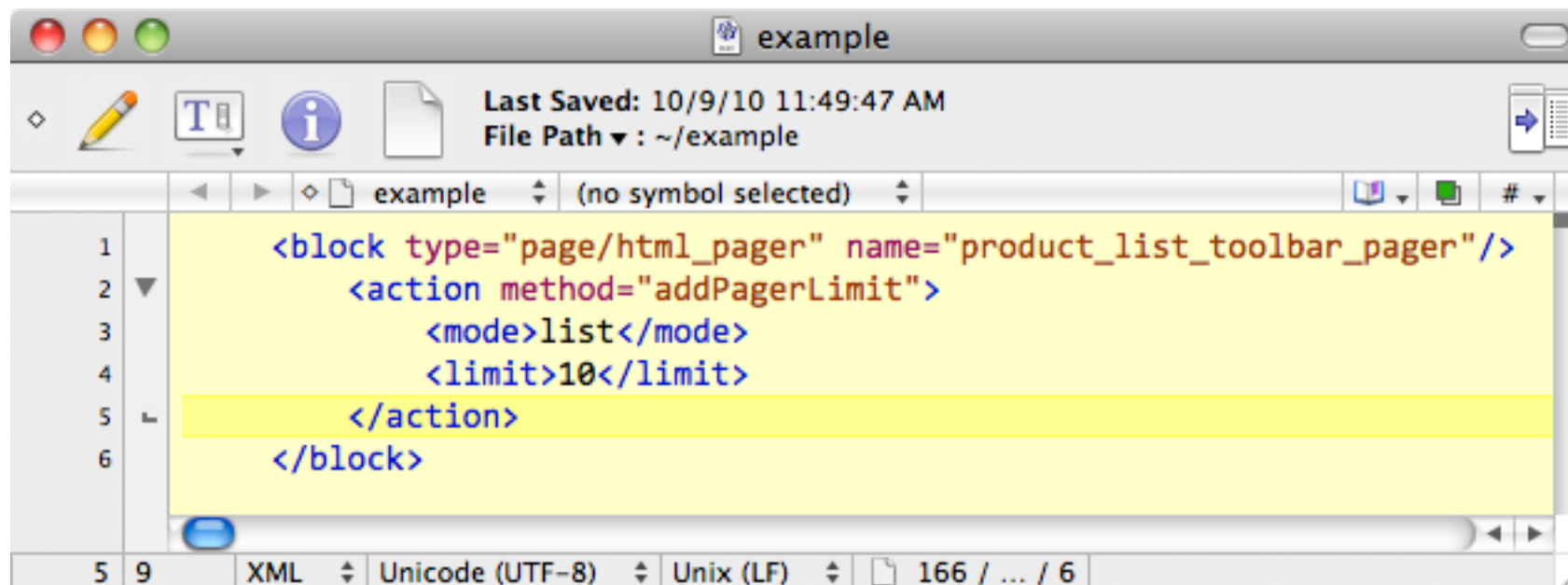
```
<reference name="content">
  <block template="catalog/category/view.phtml"
        type="catalog/category_view"
        name="category.products" >
    <block type="catalog/product_list"
          name="product_list"
          template="catalog/product/list.phtml" />
  </block>
</reference>
```

The status bar at the bottom shows '14 21 XML Unicode (UTF-8) Unix (LF) 365 / 25 / 16'.

- The **catalog/product_list** Block has been added to the **catalog/category_view** Block, with the name **product_list**
- Allows us to use the following in **catalog/category/view.phtml**
 - `$this->getChildHtml('product_list');`

Action Nodes

- Allow us to call methods on blocks from XML Layout System



The screenshot shows a text editor window titled 'example'. The editor contains the following XML code:

```
<block type="page/html_pager" name="product_list_toolbar_pager">
  <action method="addPagerLimit">
    <mode>list</mode>
    <limit>10</limit>
  </action>
</block>
```

The code is displayed with syntax highlighting. The editor's status bar at the bottom indicates the file is 166 bytes, using UTF-8 encoding and Unix (LF) line endings.

- URI page/html_pager translates to Mage_Page_Block_Html_Pager
- Calls Mage_Page_Block_Html_Pager::addPagerLimit(\$mode, \$value, \$label="")

Magento for PHP MVC Developers

Alan Storm

<http://www.magentocommerce.com/knowledge-base>

<http://alanstorm.com>

<http://twitter.com/alanstorm>

<http://commercebugdemo.alanstorm.com/>