twitter: @akrabat

Zend Framework: Next steps

Rob Allen

php|tek May 2011

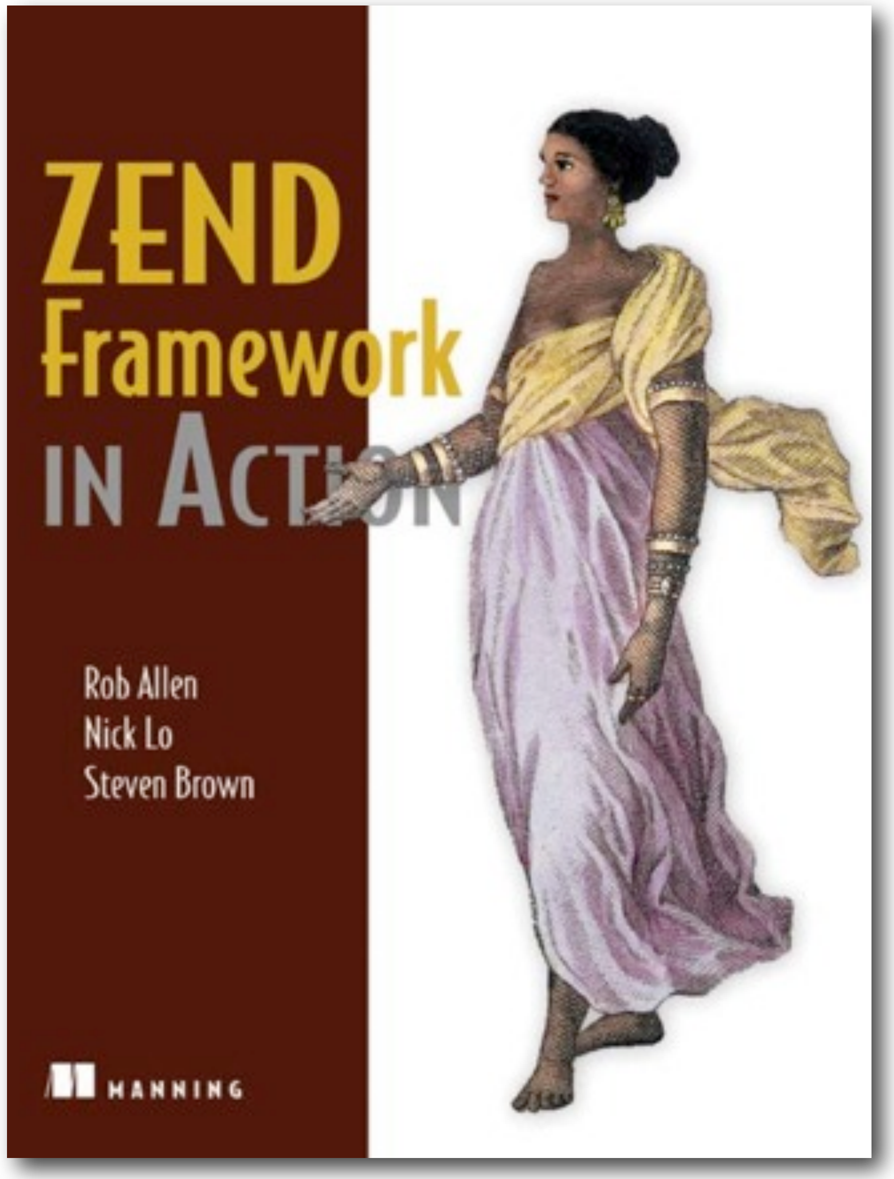
Rob Allen?

- PHP developer since 1999
- Wrote Zend_Config
- Tutorial at akrabat.com
- Zend Framework in Action!

Next book!

Zend Framework 2 in Action

(with Ryan Mauger)

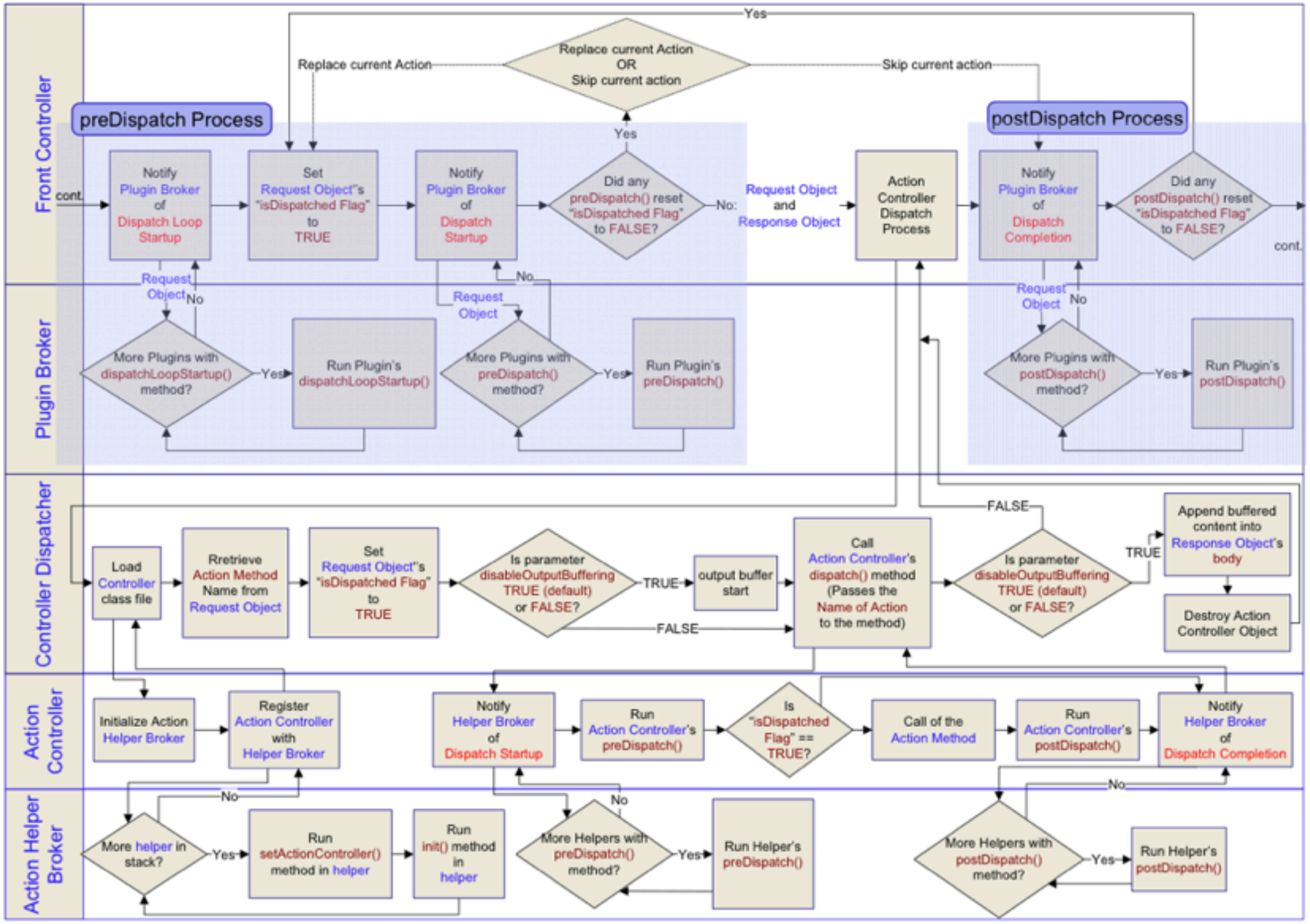


A tour through stuff I think that you you need to know!

- Dispatch cycle
- Modelling
- Authentication
- Access control
- Layouts
- Caching

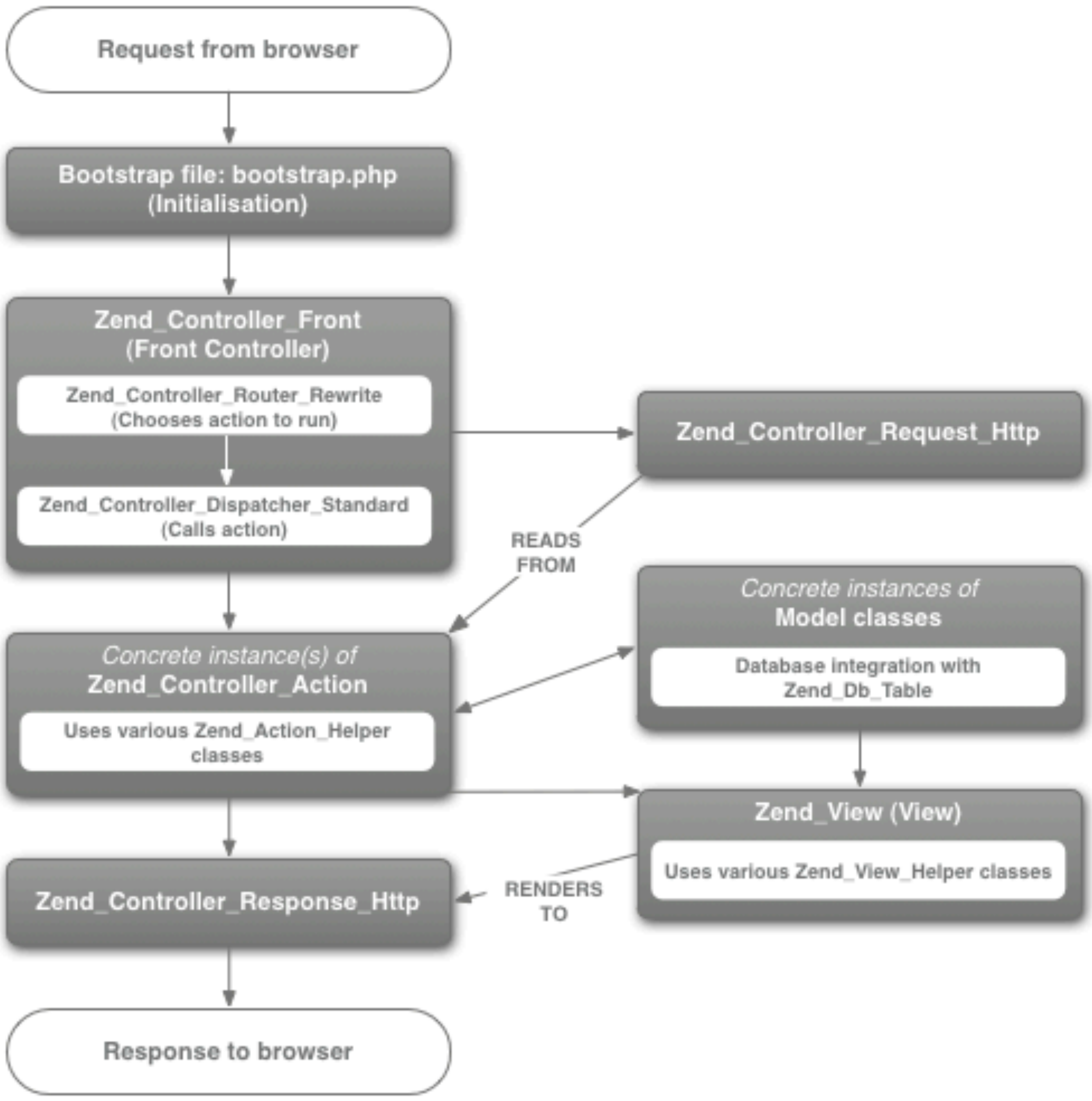
Dispatch cycle

Do you know the dispatch cycle?


Front Controller
Plugin Broker
Controller Dispatcher
Action Controller
Action Helper Broker
preDispatch Process
postDispatch Process
Replace current Action OR Skip current action
Replace current Action
Skip current action
Yes
cont.
Notify Plugin Broker of Dispatch Loop Startup
Set Request Object's "isDispatched Flag" to TRUE
Notify Plugin Broker of Dispatch Startup
Did any preDispatch() reset "isDispatched Flag" to FALSE?
No:
Request Object and Response Object
Action Controller Dispatch Process
Notify Plugin Broker of Dispatch Completion
Did any postDispatch() reset "isDispatched Flag" to FALSE?
Request Object
No
More Plugins with dispatchLoopStartup() method?
Run Plugin's dispatchLoopStartup()
More Plugins with preDispatch() method?
Run Plugin's preDispatch()
More Plugins with postDispatch() method?
Run Plugin's postDispatch()
Load Controller class file
Rretrieve Action Method Name from Request Object
Set Request Object's "isDispatched Flag" to TRUE
Is parameter disableOutputBuffering TRUE (default) or FALSE?
TRUE
FALSE
output buffer start
Call Action Controller's dispatch() method (Passes the Name of Action to the method)
Append buffered content into Response Object's body
Destroy Action Controller Object
Initialize Action Helper Broker
Register Action Controller with Helper Broker
Notify Helper Broker of Dispatch Startup
Run Action Controller's preDispatch()
Is "isDispatched Flag" == TRUE?
Call of the Action Method
Run Action Controller's postDispatch()
Notify Helper Broker of Dispatch Completion
More helper in stack?
Run setActionController() method in helper
Run init() method in helper
More Helpers with preDispatch() method?
Run Helper's preDispatch()
More Helpers with postDispatch() method?
Run Helper's postDispatch()

source: Polley Wong: http://www.slideshare.net/polleywong/zend-framework-dispatch-workflow

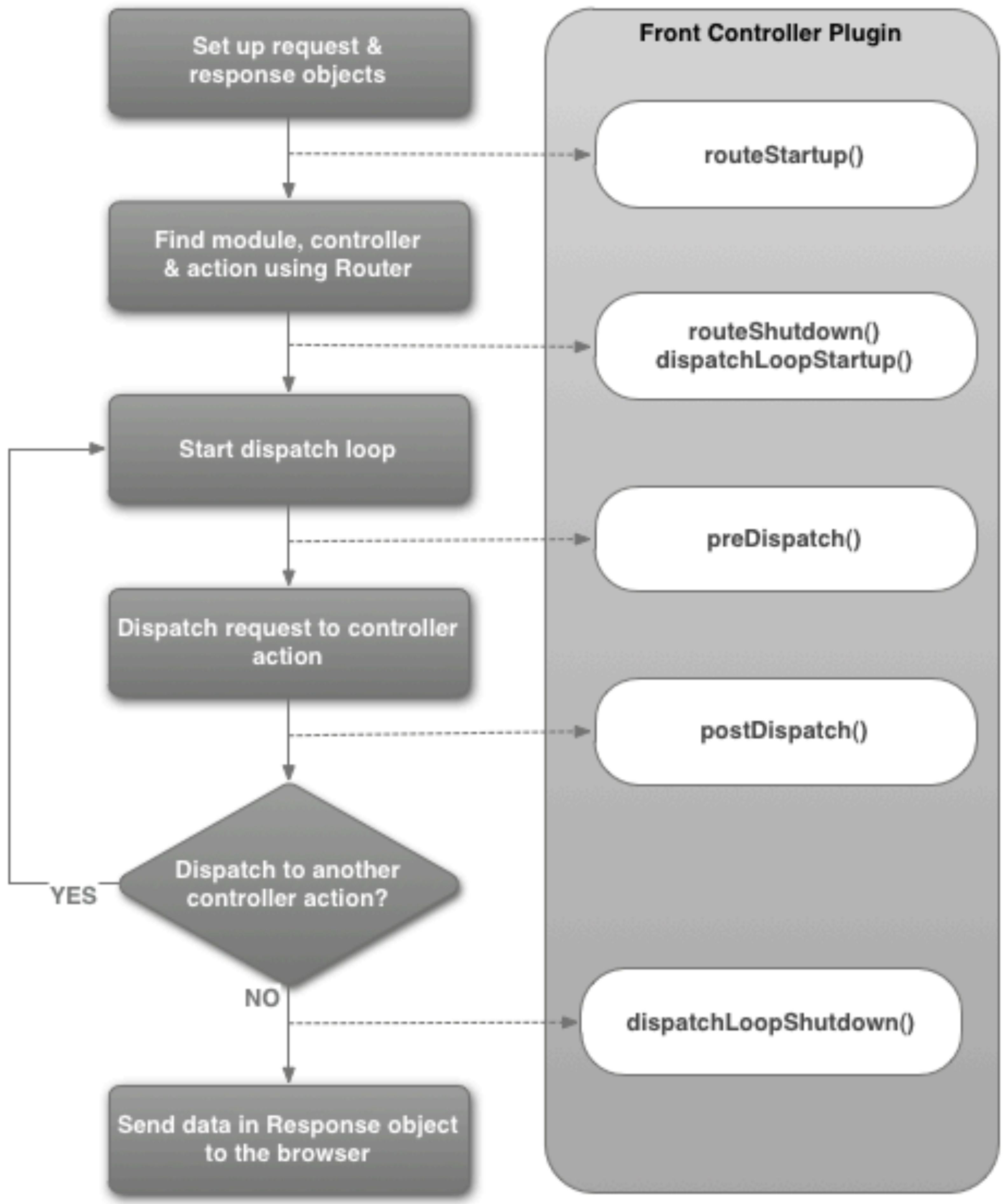
WOW!


Request from browser
Bootstrap file: bootstrap.php (Initialisation)
Zend_Controller_Front (Front Controller)
Zend_Controller_Router_Rewrite (Chooses action to run)
Zend_Controller_Dispatcher_Standard (Calls action)
Zend_Controller_Request_Http
READS FROM
Concrete instances of Model classes
Database integration with Zend_Db_Table
Concrete instance(s) of Zend_Controller_Action
Uses various Zend_Action_Helper classes
Zend_View (View)
Uses various Zend_View_Helper classes
RENDERS TO
Zend_Controller_Response_Http
Response to browser

Front Controller plugins

- Used to listen for certain events in the front controller
- Add routing / dispatch logic
- Many hook points
- Examples:
  - Layout
  - ErrorHandler

Front Controller plugins are used to add logic to the routing/dispatch process.


Set up request & response objects
Find module, controller & action using Router
Start dispatch loop
Dispatch request to controller action
Dispatch to another controller action?
YES
NO
Send data in Response object to the browser
Front Controller Plugin
routeStartup()
routeShutdown()
dispatchLoopStartup()
preDispatch()
postDispatch()
dispatchLoopShutdown()

Register

```
;application.ini
resources.frontController.plugins.acl =
Application_Plugin_Acl

// or

class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{
    public function registerAcl()
    {
        $front = Zend_Controller_Front::getInstance();
        $front->registerPlugin(new App_Plugin_Acl());
    }
}
```

Handling exceptions

- Prefer `preDispatch()` over `dispatchLoopStartup()`
- Catch the exception and modify the request to dispatch to error action
- Create an error handler object so that `errorAction()` works

Exceptions

```
public function preDispatch(Zend_Controller_Request_Abstract $request)
{
  try {
    // do something that throws an exception
  } catch (Exception $e) {
    // Repoint the request to the default error handler
    $request->setModuleName('default');
    $request->setControllerName('error');
    $request->setActionName('error');

    // Set up the error handler
    $err = new Zend_Controller_Plugin_ErrorHandler();
    $err->type = Zend_Controller_Plugin_ErrorHandler::EXCEPTION_OTHER;
    $err->request = clone($request);
    $err->exception = $e;
    $request->setParam('error_handler', $err);
  }
}
```

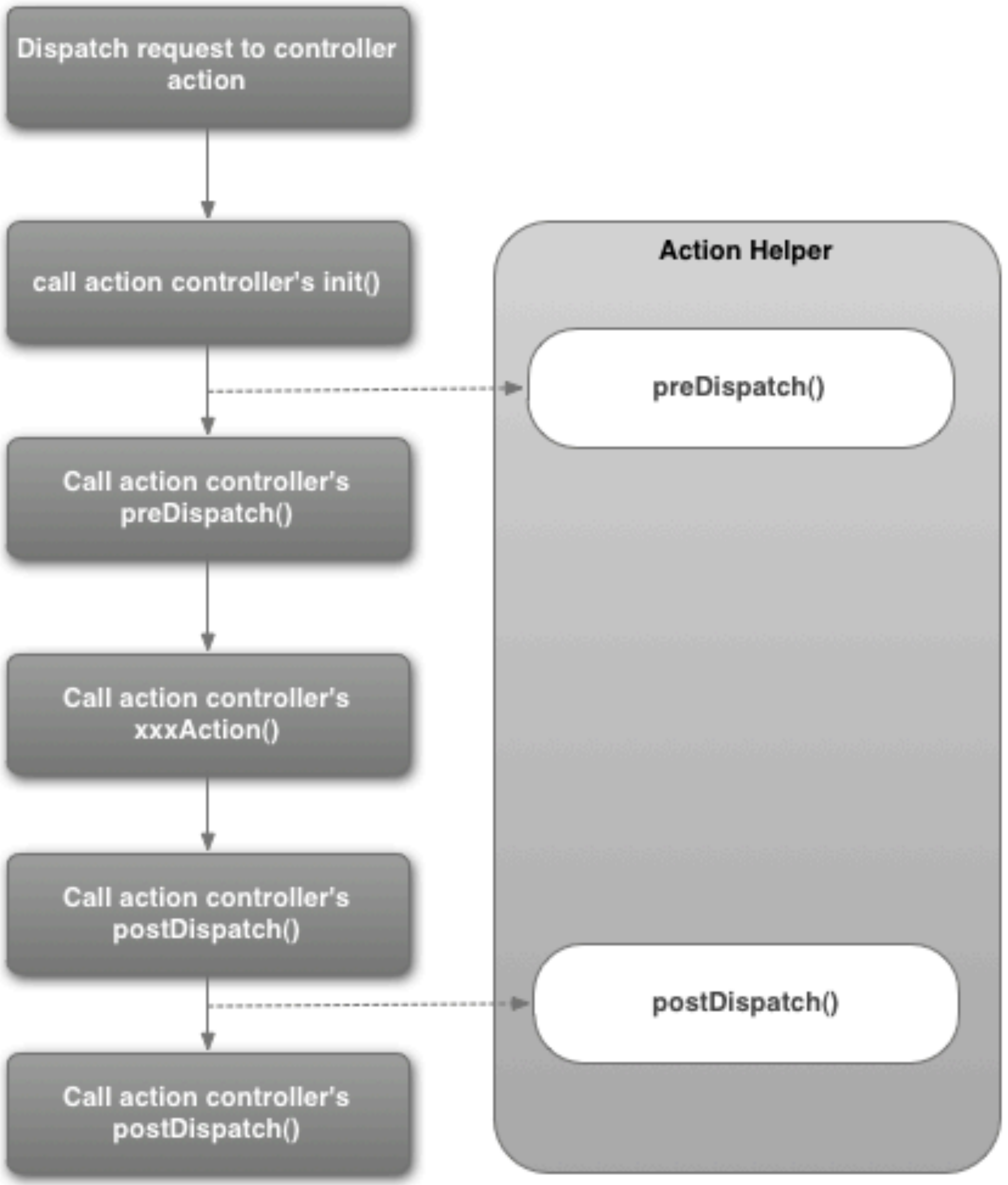
Action helpers are used to add methods to controllers.

Action helpers

- Share functionality between controllers
- Optional hooks
- Have access to the controller object

Using

```
// In a controller:
$flashMessenger = $this->_helper->FlashMessenger;
$flashMessenger->addMessage('Important message');
// OR
$this->_helper->FlashMessenger('Important message');

// Elsewhere:
use Zend_Controller_Action_HelperBroker as HBroker;
//...
$flashMessenger = HBroker::getStaticHelper('flashMessenger');
$flashMessenger->addMessage('Important message');
```


Dispatch request to controller action
call action controller's init()
Call action controller's preDispatch()
Call action controller's xxxAction()
Call action controller's postDispatch()
Call action controller's postDispatch()
Action Helper
preDispatch()
postDispatch()

Register

```
use Zend_Controller_Action_HelperBroker as HelperBroker;

// Within Bootstrap
  public function _initActionHelpers()
  {
    HBroker::addPath(APPLICATION_PATH .'/controllers/helpers');

    $quote = HelperBroker::getStaticHelper('Quote');
    HelperBroker::addHelper($quote);
  }
```

Summary

- Use front controller plugins for routing/dispatch logic
- Use action helpers to avoid repetitive code in actions

Modelling

Typical process

- Create the schema
- Write code that uses it!
  - mysqli_query()
  - ActiveRecord, Table/Row Data Gateway
- Call it our model

Problems

- Model tightly coupled to database
- Database schema based code ceases to match our API
- New features added at wrong level

How to solve?

- Models are just classes!
- Persist your models
- Expose to your application

Just a class

```
class Application_Model_Task
{
    // metadata (properties)
    protected $_title;
    protected $_due_date;
    protected $_date_completed;
    protected $_is_complete;

    // (getters and setters go here)

    // behaviour (methods)
    public function markComplete() {}

}
```

Persist your model

- Which properties?
- Where?
    - database, web service, cache, session
- How?
    - Transaction script, ActiveRecord,
      Data mapper / ORM

Persist your model

```sql
CREATE TABLE IF NOT EXISTS tasks (
  id int NOT NULL AUTO_INCREMENT,
  title varchar(200) NOT NULL,
  notes text,
  due_date datetime,
  created_by int,
  date_completed datetime,
  date_created datetime NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Persist your model

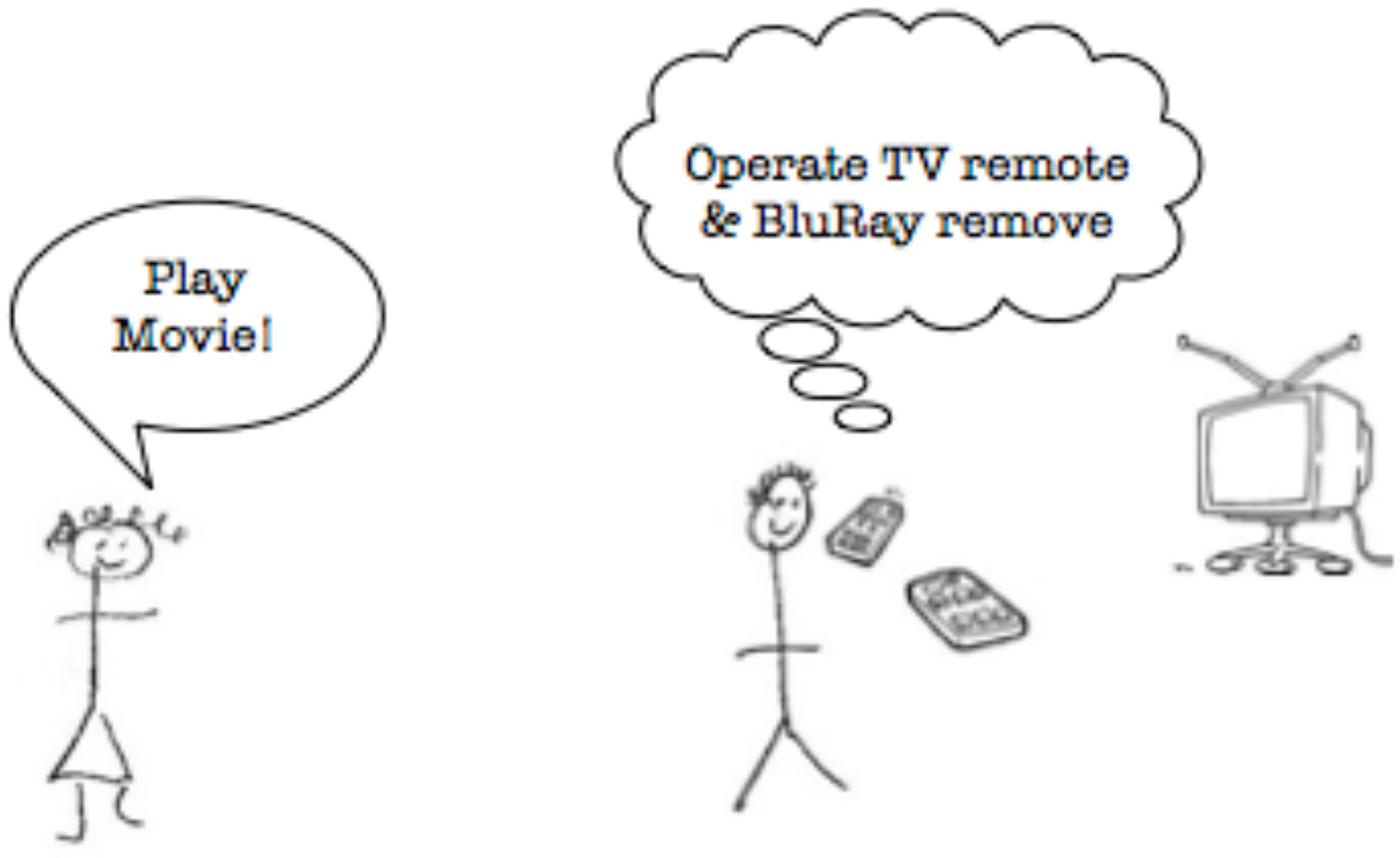
```
class Application_Model_TaskMapper
{
  public function save(Application_Model_Task $task)
  {
    $data = array(
      'title' => $task->title,
      'notes' => $task->notes,
      'due_date' => $task->due_date,
      'date_completed' => $task->date_completed,
    );

    $db = $this->getDbAdapter();
    if($task->id > 0) {
      $db->update($this->_tblName, $data, 'id = '.$task->id);
    } else {
      $db->insert($this->_tblName, $data);
    }
  }
```

Persist your model

```
public function fetchRecentlyCompleted()
{
  $db = $this->getDbAdapter();
  $select = $db->select();
  $select->from($this->_tableName);
  $select->where('date_completed IS NOT NULL');
  $select->order('date_completed DESC');
  $rows = $db->fetchAll($select);

  $tasks = array();
  foreach ($rows as $row) {
    $task = new Application_Model_Task($row);
    $tasks[] = $task;
  }
  return $tasks;
}
```

Move business and application logic to a *Service Layer*

Service layer?



Inspired by Doug Boude: http://is.gd/wcC5Ns

A *Service Layer* coordinates lower level objects to simplify the application

Data source	DBAL (Zend_Db) or web service
Domain model	Entities and mappers
Service layer	Manipulation of domain models

Why use one?

- Easy to write application via MVC layer
  - Controller maps urls to application schema
- Reuse application via services
  - (REST, AMF, SOAP, etc.)
- Reuse application via CLI scripts
- Easier to migrate to ZF2

What goes in one?

- Validation & filtering & normalisation
- Access control
- Transactions and interactions between model entities
- Caching
- Notifications (via observers?)

Service object

```
class Application_Service_TaskService
{
   public function fetchOutstanding()
    {
        $mapper = new Application_Model_TaskMapper();
        $tasks = $mapper->fetchOutstanding();
        return $tasks;
    }

    public function create(array $data)
    {
        $data = $this->processThroughInputFilter($data);
        if (!$data) {
            throw new Exception('Invalid data');
        }
        $task = new Application_Model_Task($data);
        $mapper = new Application_Model_TaskMapper();
        $mapper->save($task);
        return $task;
    }
```

ServiceLayer integration

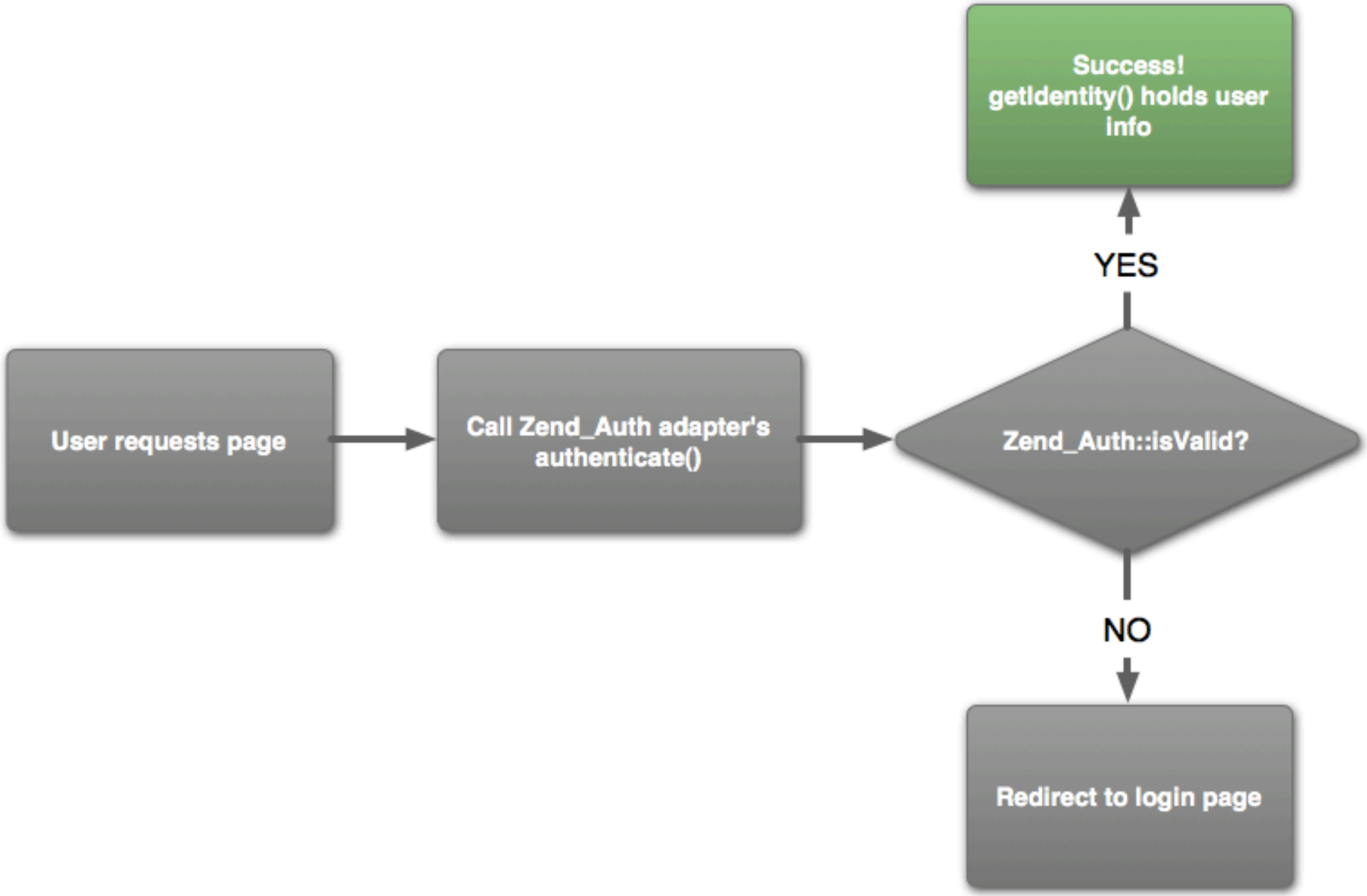
```
class IndexController extends Zend_Controller_Action
{
  public function indexAction()
  {
    $service = new Application_Service_TaskService();
    $this->view->tasks = $service->fetchOutstanding();

    $messenger = $this->_helper->flashMessenger;
    $this->view->messages = $messenger->getMessages();
  }

  // etc...
```

Summary

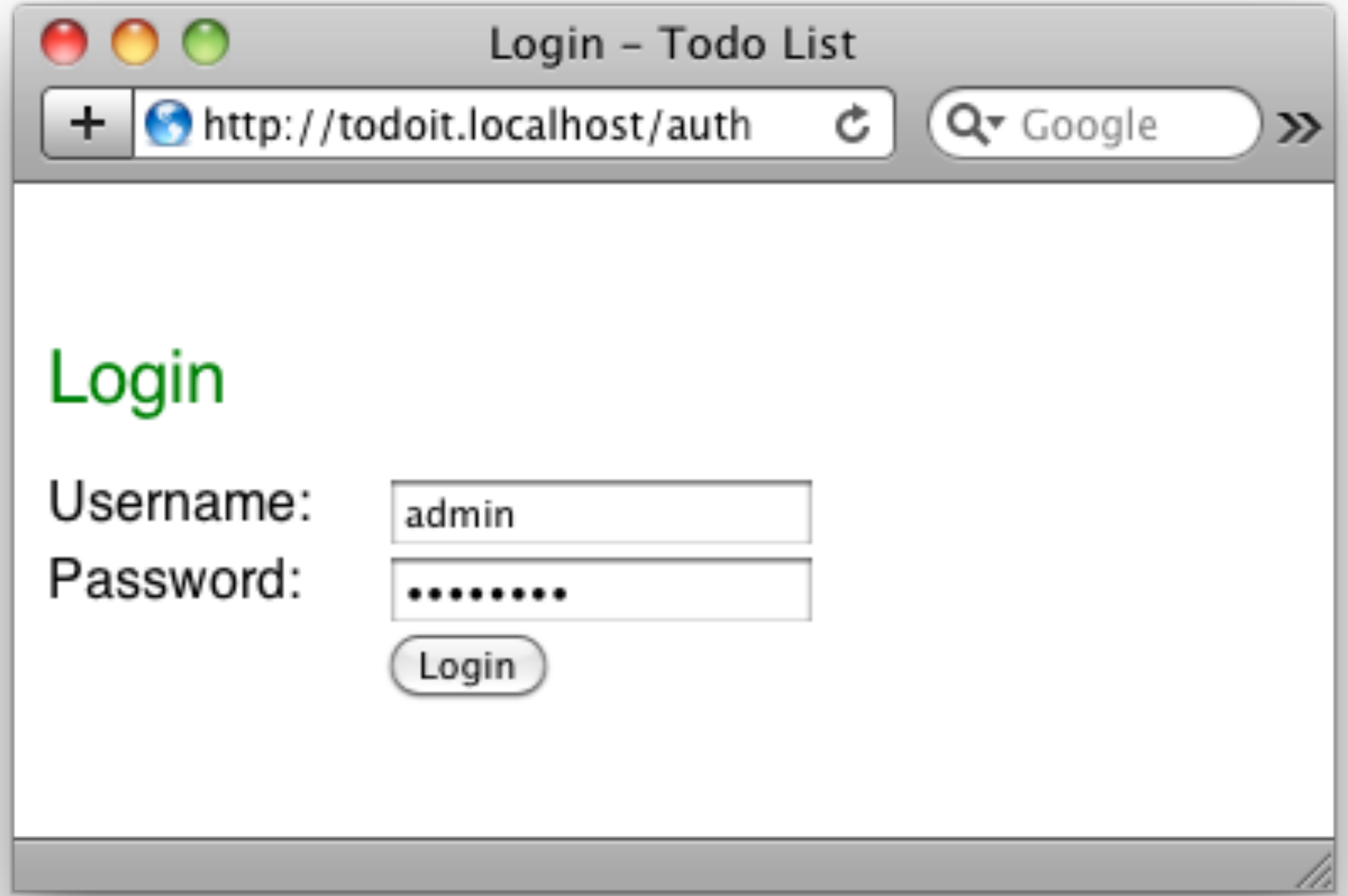
- Write your entities first.
- Separate your persistence layer so you can change it.
- Separate your business and application logic so you can refactor it on its own.
- ORMs are hard to write - use a pre-built one!

Authentication

Authentication is the process of deciding if someone is who they say they are


User requests page
Call Zend_Auth adapter's authenticate()
Zend_Auth::isValid?
YES
Success! getIdentity() holds user info
NO
Redirect to login page

Login form



Login

Username: admin

Password: ••••••••

Login form

```
class Application_Form_Login extends Zend_Form
{
  public function init()
  {
    $this->setName("login");
    $this->addElement('text', 'username', array(
        'filters' => array('StringTrim', 'StringToLower'),
        'required'   => true,
        'label'      => 'Username:',
    ));
    $this->addElement('password', 'password', array(
        'filters'    => array('StringTrim'),
        'required'   => true,
        'label'      => 'Password:',
    ));
    $this->addElement('submit', 'login', array(
        'ignore'   => true,
        'label'    => 'Login',
    ));
 }
```

AuthController

```
class AuthController extends Zend_Controller_Action
{
  public function indexAction()
  {
    $form = new Application_Form_Login();
    $request = $this->getRequest();
    if ($request->isPost()) {
      if ($form->isValid($request->getPost())) {
        if ($this->_process($form->getValues())) {
          // Success! Redirect to the home page
          $this->_helper->redirector('index', 'index');
        }
      }
    }
    $this->view->form = $form;
}
```

View script

```
<?php $this->headTitle('Login'); ?>
<h1>Login</h1>
<?php
echo $this->form->setAction($this->url());
?>
```

Authenticating

```
protected function _process($values)
{
  // Get our authentication adapter and check credentials
  $adapter = $this->_getAuthAdapter($values);
  $auth = Zend_Auth::getInstance();
  $result = $auth->authenticate($adapter);
  if ($result->isValid()) {
    $data = $adapter->getResultRowObject();
    $user = new Application_Model_User($data);
    $auth->getStorage()->write($user);
    return true;
  }
  return false;
}
```

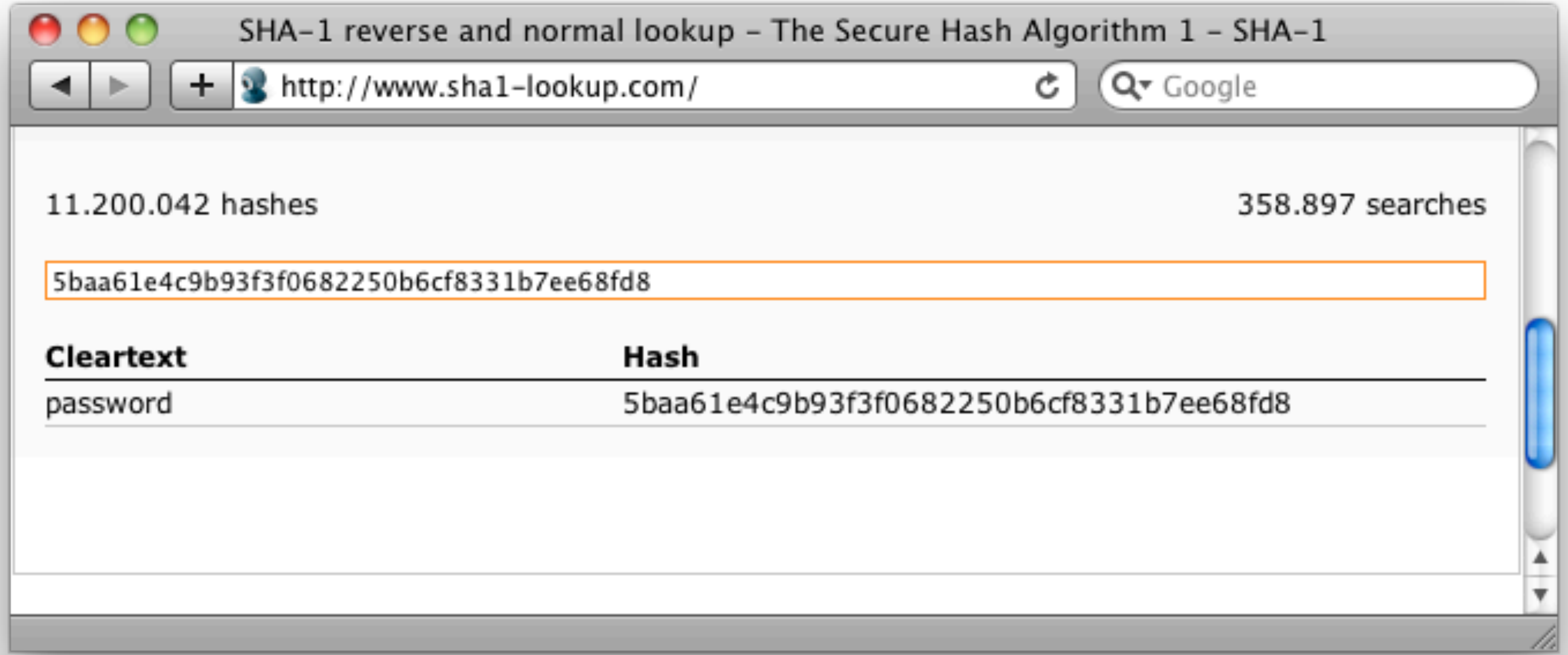
Authenticating

```
protected function _getAuthAdapter($formData) {
  $db = Zend_Db_Table::getDefaultAdapter();
  $authAdapter = new Zend_Auth_Adapter_DbTable($db);

  $authAdapter->setTableName('users')
       ->setIdentityColumn('username')
       ->setCredentialColumn('password')
       ->setCredentialTreatment('SHA1(CONCAT(?,salt))');

  $authAdapter->setIdentity($formData['username']);
  $authAdapter->setCredential($formData['password']);

  return $authAdapter;
}
```

salt?

- A random string appended to the password when storing password to the database
- Store the salt value in the users table in a separate field
- Prevents reverse SHA1 lookups

Reverse SHA1 lookup



SHA-1 reverse and normal lookup - The Secure Hash Algorithm 1 - SHA-1

http://www.sha1-lookup.com/

11.200.042 hashes

358.897 searches

5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

Cleartext	Hash
password	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

Logged in?

- Zend_Auth is a singleton, so you can do:

```
$auth = Zend_Auth::getInstance();
if ($auth->hasIdentity()) {
    $user = $auth->getIdentity();
}
```

Always force login?

```
class Application_Plugin_ForceLogin
    extends Zend_Controller_Plugin_Abstract
{
  public function dispatchLoopStartup
      (Zend_Controller_Request_Abstract $request)
  {
    $auth = Zend_Auth::getInstance();
    if (!$auth->hasIdentity()) {
      $controller = $request->getControllerName();
      if ($controller != 'auth' && $controller != 'error') {
        $redirector = Zend_Controller_Action_HelperBroker::
            getStaticHelper('redirector');
        $redirector->gotoSimple('index', 'auth');
      }
    }
  }
}
```

Summary

- Authentication is solely about logging in
- Zend_Auth has many adapters, not just database
- Always use a salt!

Access control

Authorisation is the act of determining if **somebody** has **permissions** to perform an action on a given **resource**

Roles
Resources
Rights

Roles

- Implement `Zend_Acl_Role_Interface`
- There’s one method: `getRoleId()`

User model

```php
class Application_Model_User
    implements Zend_Acl_Role_Interface
{
    protected $_role;
    // and other properties...

    public function getRoleId()
    {
        $role = $this->_role;
        return $role ? $role : 'guest';
    }
}
```

Protecting access to a controller

- Extend Zend_Acl to set up
- Use a Front Controller plugin

Extend Zend_Acl

```
class Application_Acl extends Zend_Acl
{
  public function __construct()
  {
    // Roles
    $this->addRole('guest');
    $this->addRole('user', 'guest');
    $this->addRole('administrator', 'user');

    // Resources (Controllers)
    $this->addResource(new Zend_Acl_Resource('indexController'));
    $this->addResource(new Zend_Acl_Resource('authController'));
    $this->addResource(new Zend_Acl_Resource('errorController'));

    // Rules for controller access
    $this->deny();
    $this->allow('guest', 'authController', null);
    $this->allow('guest', 'errorController', null);
    $this->allow('user', 'indexController', null);
  }
```

Front controller plugin

```
// application/plugins/Acl.php
class Application_Plugin_Acl
    extends Zend_Controller_Plugin_Abstract
{
    public function dispatchLoopStartup (
        Zend_Controller_Request_Abstract $request)
    {
    }
}

; application/configs/application.ini
resources.frontController.plugins.acl =
    Application_Plugin_Acl
```

dispatchLoopStartup

```php
public function dispatchLoopStartup(Zend_Controller_Request_Abstract $request)
{
  $acl = $this->getAcl(); /* @var $acl Application_Acl */
  $user = $this->getCurrentUser();
  $resource = $request->getControllerName() . 'Controller';
  $privilege = $request->getActionName();

  $allowed = $acl->isAllowed($user, $resource, $privilege);
  if (!$allowed) {
    $controller = 'auth';
    $auth = $this->getAuth();
    if (!$auth->hasIdentity()) {
      $action = 'index';
    } else {
      $action = 'permissions';
    }
    $r = Zend_Controller_Action_HelperBroker::getStaticHelper('redirector');
    $r->gotoSimple($action, $controller);
  }
}
```

Get current role

```
public function getCurrentUser()
{
  if (!$this->_currentUser) {
    $auth = Zend_Auth::getInstance();
    if ($auth->hasIdentity()) {
      $this->_currentUser = $auth->getIdentity();
    } else {
      $this->_currentUser = new Application_Model_User();
    }
  }
  return $this->_currentUser;
}
```

Protecting models

- Implement `Zend_Acl_Resource_Interface`
    - One method: `getResourceId()`
- Use a *ServiceLayer* to do the ACL work
- Use ACL assertions too!

Add to model

```
class Application_Model_Task
    implements Zend_Acl_Resource_Interface
{
    // ... other methods ...

    public function getResourceId()
    {
        return 'task';
    }

    // ... other methods ...
}
```

ServiceLayer integration

```
class Application_Service_TaskService
{
  public function fetchOutstanding()
  {
    $acl = $this->getAcl();
    $user = $this->getCurrentUser();
    $mapper = new Application_Model_TaskMapper();
    $tasks = $mapper->fetchOutstanding();
    foreach ($tasks as $i => $task) {
      if (!$acl->isAllowed($user, $task, 'read')) {
          unset($tasks[$i]);
      }
    }
    return $tasks;
  }
```

ServiceLayer integration

```
public function getAcl ()
{
  if (! $this->_acl) {
    $this->_acl = new Application_Acl();
    $this->_acl->allow('user', 'task',
        array('read', 'update', 'delete'),
        new Application_Model_Acl_AssertUserOwnsTask());
    $this->_acl->allow('user', 'task', 'create');
    $this->_acl->allow('administrator', 'task',
        array('read', 'update', 'delete'));
  }
  return $this->_acl;
}
```

Acl assertion

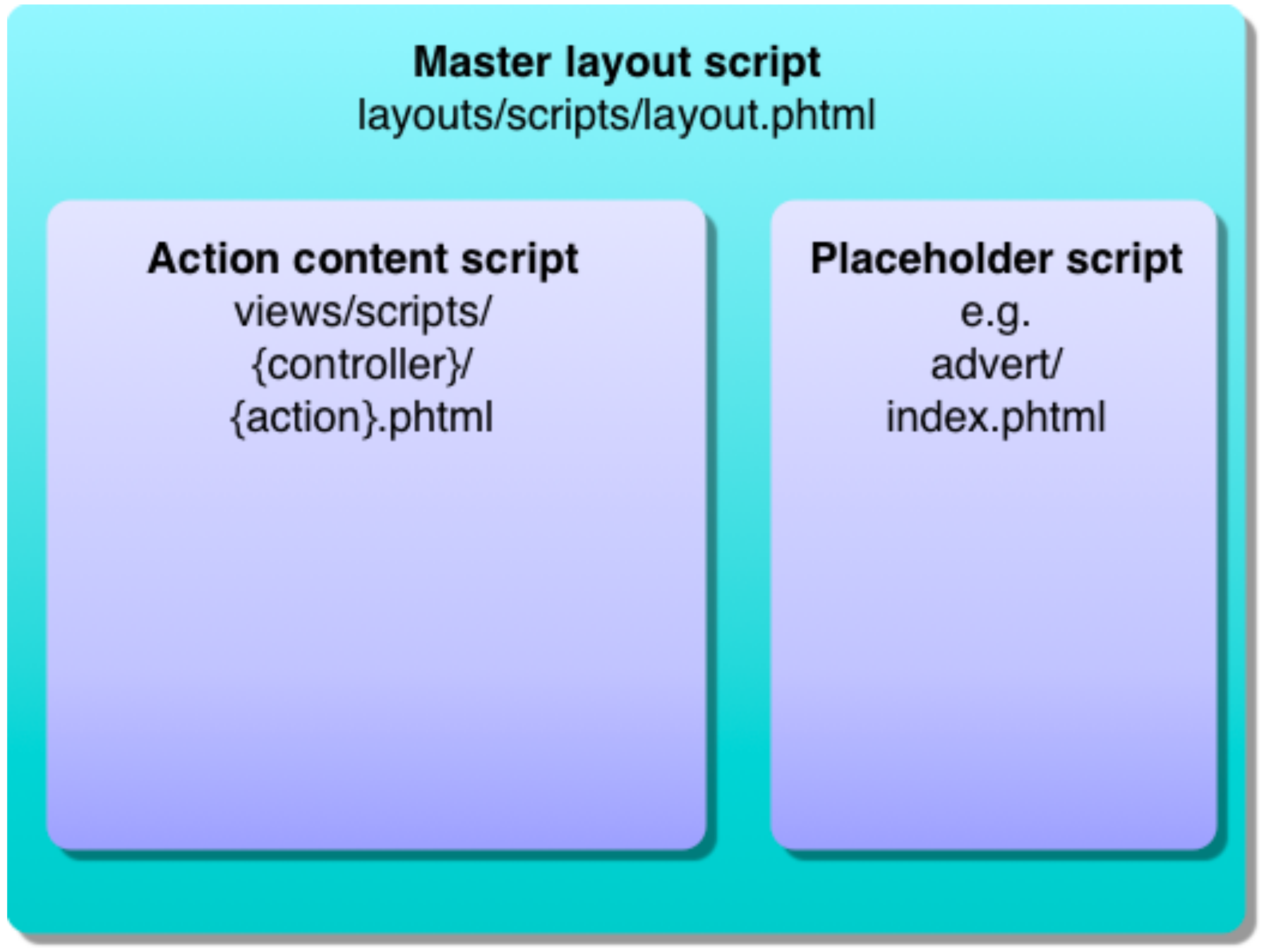
```
class Application_Model_Acl_AssertUserOwnsTask
    implements Zend_Acl_Assert_Interface
{
  public function assert(Zend_Acl $acl,
          Zend_Acl_Role_Interface $role = null,
          Zend_Acl_Resource_Interface $resource = null,
          $privilege = null)
  {
    $auth = Zend_Auth::getInstance();
    if (!$auth->hasIdentity()) {
      return false;
    }
    $user = $auth->getIdentity();

    return $resource->getCreatedBy() == $user->getId();
  }
}
```

Summary

- `Zend_Acl` is very flexible!
- Integrate with your objects using the available `Zend_Acl` interfaces
- Use dynamic assertions when appropriate

Layouts

Composite View



Simple layout script

```
<?php echo $this->doctype() ?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <?php echo $this->headMeta(); ?>
    <?php echo $this->headTitle(); ?>
    <?php echo $this->headLink(); ?>
    <?php echo $this->headScript(); ?>
</head>
<body>
    <?php echo $this->layout()->content ?>
    <?php echo $this->inlineScript(); ?>
</body>
</html>
```

Layout view helpers

- Set metadata, JS and CSS within action view scripts
- Aggregate content
- Rendered in the layout

doctype()

Setting:

```
; application.ini
resources.view.doctype = HTML5
```

Rendering:

```
<!-- in layouts/scripts/layout.phtml -->
<?php echo $this->doctype(); ?>
```

headXxx()

- Used for setting all <head> tags
- Can append and prepend
- Available:
    - `headMeta()`
    - `headTitle()`
    - `headScript()`
    - `headStyle()`

in layout.phtml

```
<?php
$this->headMeta()->appendHttpEquiv(
   'Content-Type', 'text/html;charset=utf-8');
$this->headTitle('Todo List')->setSeparator(' - ');
$this->headLink()->prependStylesheet(
    $this->baseUrl('/css/site.css'));

echo $this->doctype() ?>
<html>
<head>
    <?php echo $this->headMeta(); ?>
    <?php echo $this->headTitle(); ?>
    <?php echo $this->headLink(); ?>
    <?php echo $this->headScript(); ?>
</head>
```

Generated HTML

```html
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type"
        content="text/html;charset=utf-8" >
    <title>Todo List</title>
    <link href="/css/site.css" media="screen"
        rel="stylesheet" type="text/css" >
</head>
<body>
<!-- etc -->
```

inlineXxx()

- `inlineStyle()`
  - Use for setting inline CSS within `<head>`
- `inlineScript()`
  - Use for inline JS just before `</body>`

Many layout scripts?

```
class Zend_View_Helper_HeadSection
    extends Zend_View_Helper_Abstract
{
  public function headSection()
  {
     $view = $this->view;
     $view->headMeta()->appendHttpEquiv(
       'Content-Type', 'text/html;charset=utf-8');
     $view->headTitle('Todo List')
       ->setSeparator(' - ');
     $html = $view->headMeta();
     $html .= $view->headTitle();
     return $html;
  }
}
```

All layout script files

```
<?php echo $this->doctype(); ?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <?php echo $this->headSection(); ?>
</head>
```

Action specific

- Some actions need JS/CSS included in `<head>`

  (or JS just before `</body>`)
- Place this code in the action’s view script

in add.phtml

```
<?php $this->headTitle('Add task');
$this->headLink()->appendStylesheet(
    $this->baseUrl('/css/product.css'), 'screen');
?>
<h1>Add task</h1>
<?php
$this->form->setAction($this->url());
echo $this->form;
?>
```

Generated HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type"
        content="text/html;charset=utf-8" >
    <title>Add task - Todo List</title>
    <link href="/css/site.css" media="screen"
        rel="stylesheet" type="text/css" >
    <link href="/css/product.css" media="screen"
        rel="stylesheet" type="text/css" >
</head>
<body>
<!-- etc -->
```

Inline JavaScript

```
<?php
$script = <<<EOT
$(document).ready(function() {
    $("a[rel^='prettyPhoto']").prettyPhoto();
    $(".toolbar a").hover(function() {
        $(this).addClass('ui-state-hover');
    }, function() {
     $(this).removeClass('ui-state-hover');
    });
});
EOT;
$this->inlineScript()->appendScript($script);
?>
```

Render in layout

```
    <!-- ...  -->
    <?php echo $this->layout()->content ?>
    <?php echo $this->inlineScript(); ?>
</body>
</html>
```

Summary

- Control your <head> tags from your view layer
- set doctype()!
- Don’t forget to add all headXxx() and inlineXxx() methods to the layout.phtml

Caching

Measure! Measure! Measure!

(siege, ab or wcat)

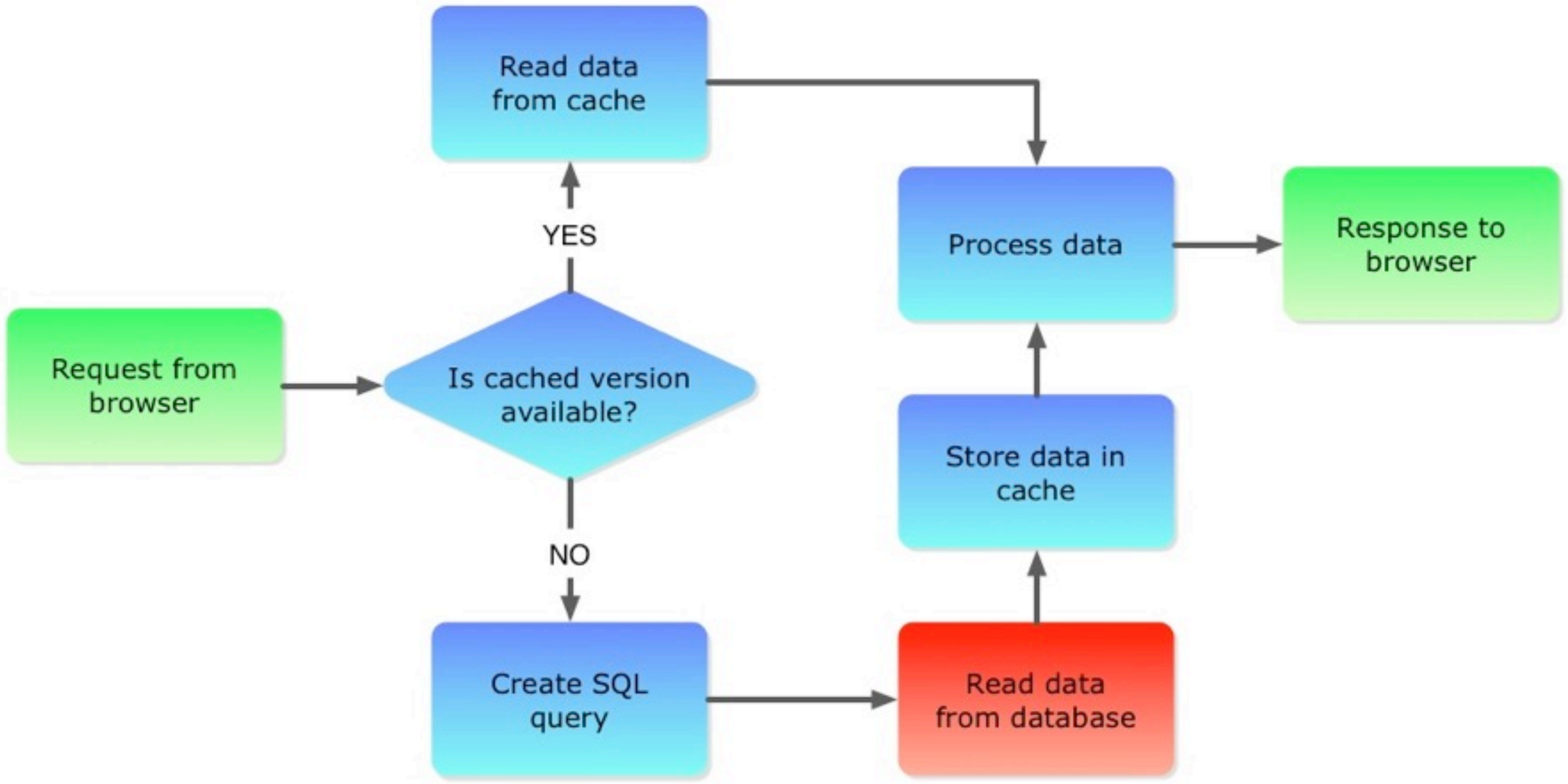
Profile! Profile! Profile!

(Xdebug or Zend Studio, Firebug)

Principles of caching

1. Don’t execute code unless you need to
2. Don’t get the same data twice
3. Get data from the fastest place you can

Code caching



Zend_Cache

- Front end adapters
  - **What** to cache
- Back end adapters
  - **Where** to cache

Zend_Cache

- Front end adapters
  - `Zend_Cache_Core`
  - `Zend_Cache_Frontend_Output`
  - `Zend_Cache_Frontend_Function`
  - `Zend_Cache_Frontend_Class`
  - `Zend_Cache_Frontend_File`
  - `Zend_Cache_Frontend_Page`

Zend_Cache

- Backends
  - `Zend_Cache_Backend_File`
  - `Zend_Cache_Backend_Sqlite`
  - `Zend_Cache_Backend_Memcached`
  - `Zend_Cache_Backend_Apc`
  - `Zend_Cache_Backend_Xcache`
  - `Zend_Cache_Backend_ZendServer`
  - `Zend_Cache_Backend_Static`

Zend_Cache_Manager

- Manages multiple cache objects
- Lazy loads on demand
- Contains preconfigured caches
- Application resource for creation
- Action helper for access

Current code

```
class Application_Service_TaskService
{
   public function fetchOutstanding()
    {
         $mapper = new Application_Model_TaskMapper();
         $tasks = $mapper->fetchOutstanding();
         return $tasks;
    }
```

Adding a cache

1. Set up the cache
2. Wrap cache loading around current code
3. Clear cache on data change

Set up

```
; application.ini - in [production]

resources.cachemanager.default.frontend
    .options.caching = 1
resources.cachemanager.default.frontend
    .options.lifetime = 7200
resources.cachemanager.default.frontend
    .options.automatic_serialization = true
resources.cachemanager.default.backend
    .options.cache_dir = APPLICATION_PATH "/../var/cache"
```

Retrieve in a controller

```
class IndexController extends Zend_Controller_Action
{
    public function indexAction()
    {
        $this->_helper->getHelper('Cache')
             ->getManager()
             ->getCache('default');
    }
}
```

Retrieve in service

```
class Application_Service_TaskService
{
    protected function _getCache($cache = 'default')
    {
        if (! $this->_cache) {
            $fc = Zend_Controller_Front::getInstance();
            $cache = $fc->getParam('bootstrap')
                        ->getResource('cachemanager');
                        ->getCache($default);
            $this->_cache = $cache;
        }
        return $this->_cache;
    }
```

Wrap current code

```
class Application_Service_TaskService
{
  public function fetchOutstanding()
  {
    $cacheId = 'outstandingTasks';
    $cache = $this->_getCache();
    $tasks = $cache->load($cacheId);
    if ($tasks === false) {
      $mapper = new Application_Model_TaskMapper();
      $tasks = $mapper->fetchOutstanding();
      $cache->save($tasks, $cacheId, array('tasks'));
    }
    return $tasks;
  }
```

Unique id

Existing

Tag

Store to cache

Emptying the cache

```
class Application_Service_TaskService
{
    protected function _cleanCache()
    {
        $this->_getCache()
            ->clean(Zend_Cache::CLEANING_MODE_MATCHING_TAG,
                array('tasks'));
    }

    public function _cleanAllCache()
    {
        return $this->_getCache()
                   ->clean(Zend_Cache::CLEANING_MODE_ALL);
    }
```

Summary

- Store to fastest back-end.
- Install APC
- Configure HTTP headers

Summary

Never stop learning!

Questions?

feedback: http://joind.in/3459

email: rob@akrabat.com

twitter: @akrabat

Thank you

feedback: http://joind.in/3459
email: rob@akrabat.com
twitter: @akrabat