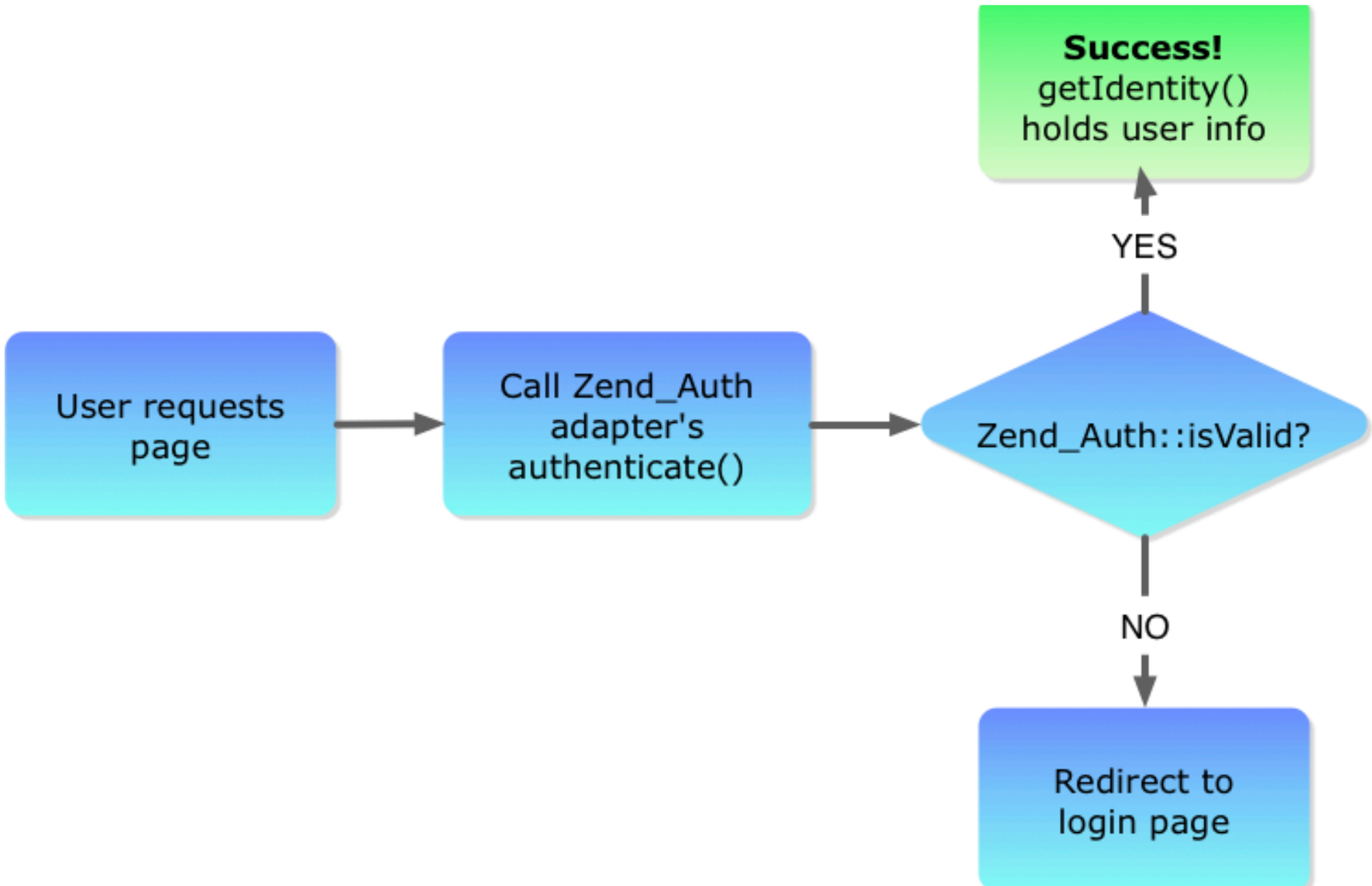
ZF authentication and access control

Rob Allen

Authentication

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

Zend_Auth process



AuthController

```
$ zf create controller Auth
Creating a controller at /www/todolist/application/
controllers/AuthController.php
Creating an index action method in controller Auth
Creating a view script for the index action method at /www/
todolist/application/views/scripts/auth/index.phtml
Creating a controller test file at /www/todolist/tests/
application/controllers/AuthControllerTest.php
Updating project profile '/www/todolist/.zfproject.xml'

$ zf create form Login
Creating a form at /www/todolist/application/forms/Login.php
Updating project profile '/www/todolist/.zfproject.xml'
```

Users table

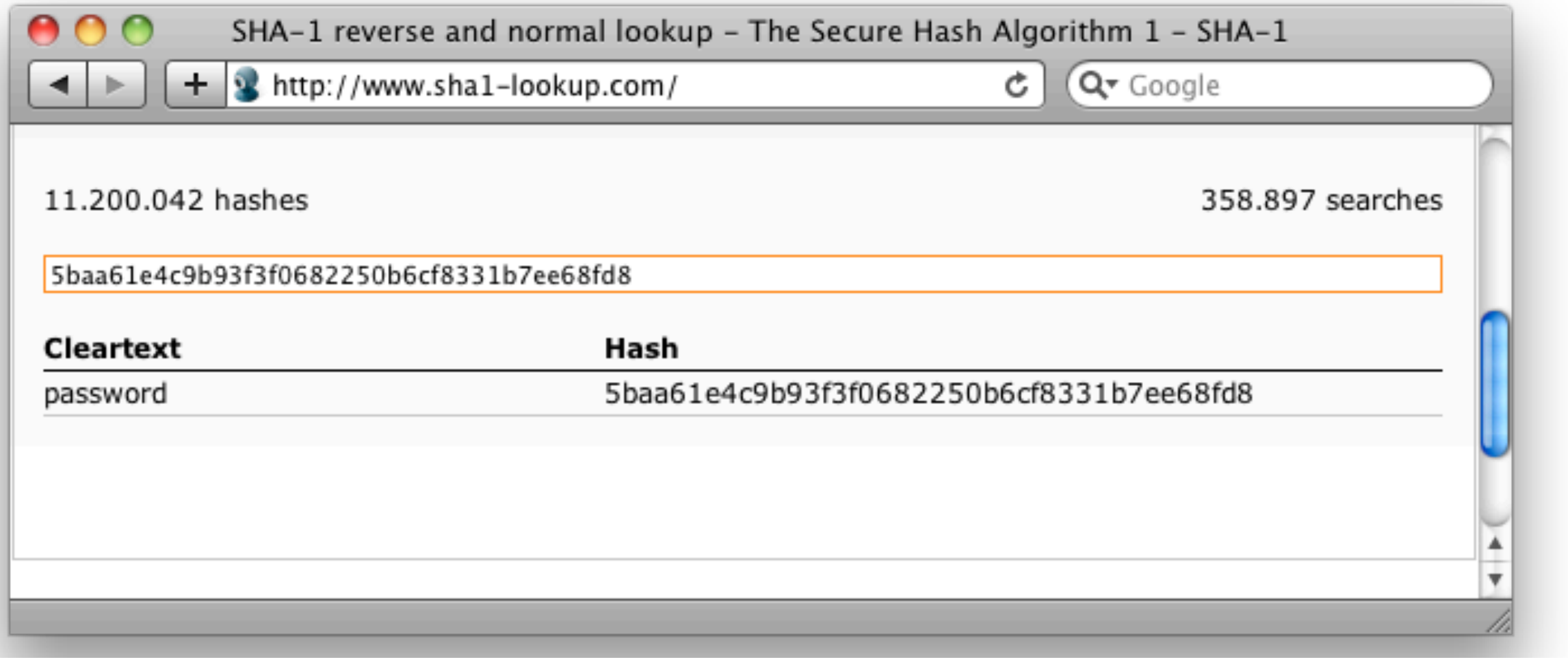
```
CREATE TABLE IF NOT EXISTS users (
  id int NOT NULL AUTO_INCREMENT,
  username varchar(50) NOT NULL,
  password varchar(50) NOT NULL,
  date_created datetime NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- insert first user
INSERT INTO users (username, password, date_created) VALUES
('admin', SHA1('AkR654_password'), NOW());
```

application.ini

```
auth.salt = "AkR654_"
```

Reverse SHA1 lookups

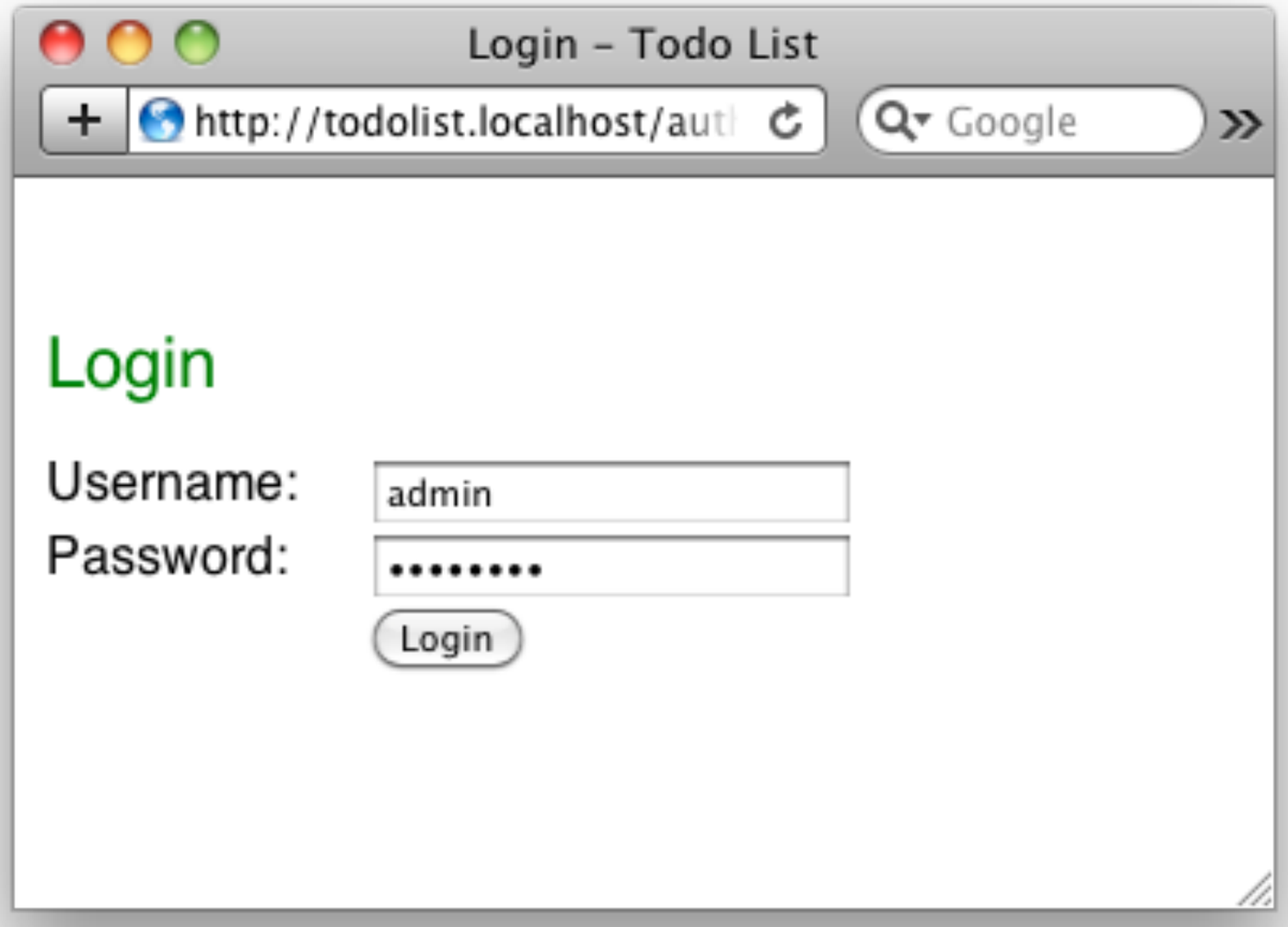
```
SHA1('password') = 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
SHA1('AkR654_password') = b871d8401e775fa80c15f40186064c0a632201f7
```



Login form

```
class Application_Form_Login extends Zend_Form
{
    public function init()
    {
        $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(
            'required' => false,
            'ignore'   => true,
            'label'    => 'Login',
        ));
    }
}
```

Login form


Login – Todo List
http://todolist.localhost/aut
Google
Login
Username:
admin
Password:
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;
    }
```

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) {
    $dbAdapter = Zend_Db_Table::getDefaultAdapter();
    $authAdapter = new Zend_Auth_Adapter_DbTable($dbAdapter);
    $authAdapter->setTableName('users')
        ->setIdentityColumn('username')
        ->setCredentialColumn('password')
        ->setCredentialTreatment('SHA1(?)');

    $fc = Zend_Controller_Front::getInstance();
    $options = $fc->getParam('bootstrap')->getOptions();
    $password = $options['auth']['salt'].$formData['password'];

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

    return $authAdapter;
}
```

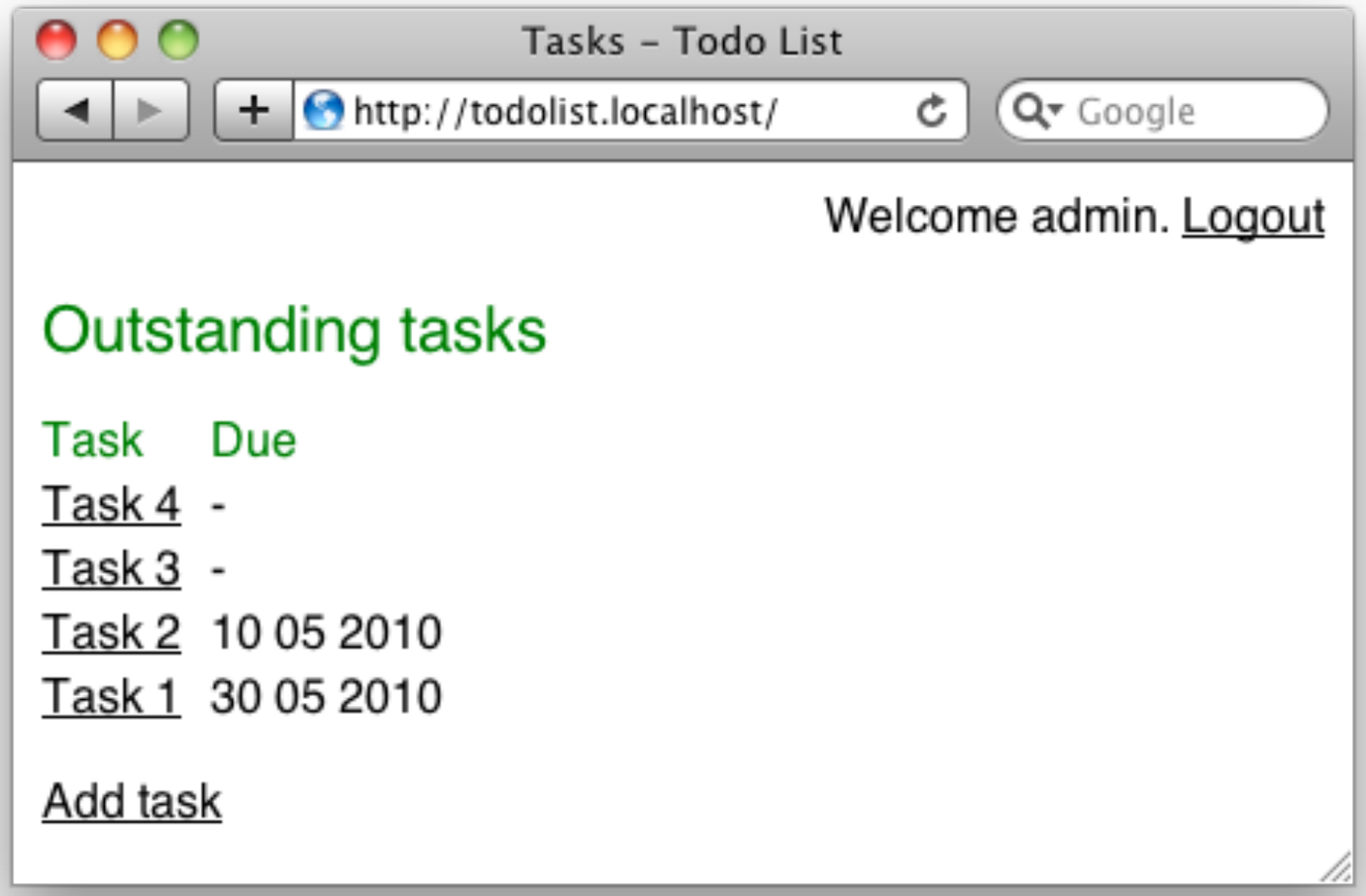
LoggedInAs view helper

```
class Zend_View_Helper_LoggedInAs extends Zend_View_Helper_Abstract
{
  public function loggedInAs ()
  {
    $auth = Zend_Auth::getInstance();
    if ($auth->hasIdentity()) {
      $username = $auth->getIdentity()->getUsername();
      $url = $this->view->url(array('controller'=>'auth',
          'action'=>'logout'), null, true);
      return 'Welcome '.$username.'<a href="'.$url.'">Logout</a>';
    }

    $url = $this->view->url(array('controller'=>'auth',
          'action'=>'index'));
    return '<a href="'.$url.'">Login</a>';
  }
}
```

Layout.phtml

```
<div id="header">
    <div id="logged-in-as">
        <?php echo $this->loggedInAs(); ?>
    </div>
</div>
```

LoggedInAs in action



Access control

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

The three "R"s

- **Roles** - what/who requests the action
- **Resources** - what is being acted upon
- **Rights** - the privileges a role has for a given resource

Roles

- Implement *Zend_Acl_Role_Interface*
- One method: *getRoleId()*

User model

```
class Default_Model_User implements Zend_Acl_Role_Interface
{
  protected $_role = 'administrator';

  public function getRoleId()
  {
    return $this->getRole();
  }
}
```

Protecting controllers

- 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'));

    }
}
```

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
```

Acl plugin

```
public function dispatchLoopStartup(Zend_Controller_Request_Abstract $request)
{
    $acl = $this->getAcl();
    $role = $this->getCurrentUser();
    $resource = $request->getControllerName() . 'Controller';
    $privilege = $request->getActionName();

    $allowed = $acl->isAllowed($role, $resource, $privilege);
    if (!$allowed) {
        $controller = 'auth';
        $auth = $this->getAuth();
        if (!$auth->hasIdentity()) {
            $action = 'index';
        } else {
            $action = 'permissions';
        }
        $redirector = new Zend_Controller_Action_Helper_Redirector();
        $redirector->gotoSimpleAndExit($action, $controller);
    }
}
```

Setup ACL rules

```
public function getAcl()
{
    if (null === $this->_acl) {
        $acl = new Application_Acl();

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

        $this->_acl = $acl;
    }
    return $this->_acl;
}
```

Role Resource Privileges

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 your models

- Implement *Zend_Acl_Roesource_Interface*
  - One method: *getResourceId()*
- Use a *ServiceLayer* to do the ACL work

Add to model

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

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

}
```

ServiceLayer integration

```
class Application_Model_TaskService
{
    public function fetchOutstanding()
    {
        $acl = $this->getAcl();
        $user = $this->getCurrentUser();
        if (!$acl->isAllowed($user, 'task', 'read')) {
            throw new Exception("Not Allowed");
        }

        $cacheId = 'outstandingTasks';
        $cache = $this->_getCache();
        $rows = $cache->load($cacheId);
        if ($rows === false) {
            $tasksGateway = new Application_Model_DbTable_Tasks();
            $rows = $tasksGateway->fetchOutstanding();
            $cache->save($rows, $cacheId, array('tasks'));
        }
        return $rows;
    }
}
```

ServiceLayer integration

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

}
```

Thank you