pawel.rajba@gmail.com, http://kursy24.eu/

Zend Framework

Plan wykładu

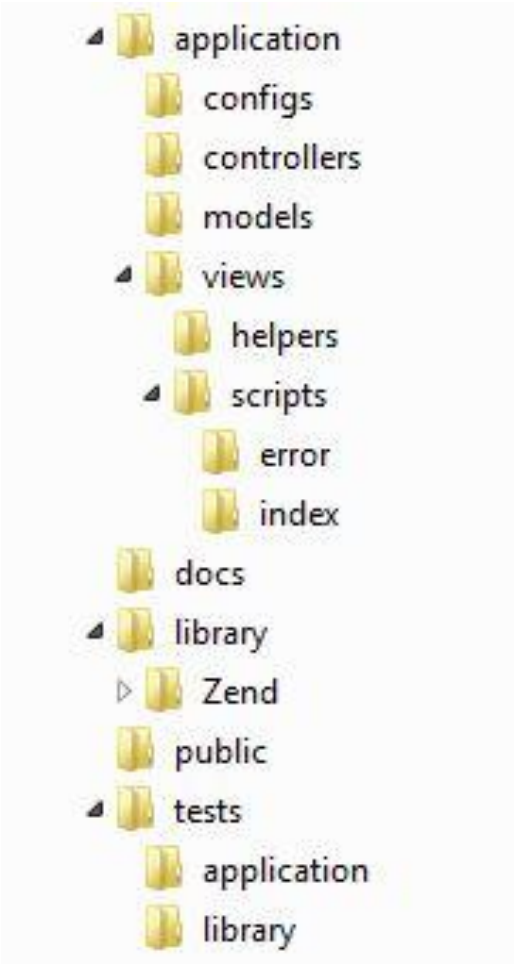
- Zend Framework
  - Wprowadzenie
  - Funkcjonalności
  - Adresy i nawigacja
  - Obiekty request i response
  - Ciastka
  - Layout, kontroler, widok
  - Sesje
  - Formularze
  - Uwierzytelnianie i autoryzacja
  - Dostęp do danych

Wprowadzenie

- Dostęp za free
- Rozwijany przez firmę Zend
- Wspiera tworzenie aplikacji w wielu aspektach:
  - Sesje
  - Formularze wraz z walidacją
  - Filtry
  - Uwierzytelnianie i autoryzacja
  - Obsługa danych
- Tworzenie aplikacji wspiera Zend Studio

Wprowadzenie

- Struktura plików



- Przykład:
  - Structure

Wprowadzenie

- Zanim przejdziemy dalej…
  - W pliku /public/.htaccess możemy wpisać wiersz SetEnv APPLICATION_ENV development
    - określi on sposób zachowania podczas uruchomienia i prezentacji w przeglądarce
    - dostępne opcje: production, development, testing
    - zachowanie w każdym przypadku jest w określone w pliku /application/configs/application.ini
    - powyższe można także ustawić w pliku /public/index.php

Wprowadzenie

- Zanim przejdziemy dalej…
  - W opublikowanej aplikacji nie jest dobrze, jeśli w przypadku błędu pojawiają się jakieś „blue screeny”
  - Należy utworzyć ErrorController oraz odpowiedni widok i tam sformatować stronę w przypadku błędu
  - Tworząc projekt w Zend Studio powyższe tworzone jest automatycznie

Funkcjonalności

- Omówione zostaną następujące możliwości:
  - Adresy i nawigacja
  - Obiekty request i response
  - Ciastka
  - Tworzenie layoutu, kontrolera i widoków
    - Helpery do kontrolera
    - Helpery do widoków i mechanizm widoków częściowych
  - Sesje
  - Formularze i ich walidacja
  - Uwierzytelnianie i autoryzacja
  - Dostęp do danych
    - Wbudowany mechanizm Zend Framework
    - Integracja z Doctrine

Adresy i nawigacja

- Domyślny schemat adresowania:
  - Controller/Action/Var1/Val1/Var2/Val2/...
    - ten domyślny schemat można zmienić tworząc „routes” więcej: http://framework.zend.com/manual/en/zend.controller.router.html
  - Tworząc linki pomiędzy stronami należy korzystać z odpowiednich „helperów”

Adresy i nawigacja

- Po stronie kontrollera mamy przekierowania:
  - $this->_helper->redirector(akcja,kontroler,moduł,parametry)
  - $this->_redirect(url,opcje)
  - $this->_forward(akcja,kontroler,moduł,parametry)
    - Nie trzeba podawać wszystkich parametrów, przykład
      - $this->_helper->redirector("login","auth");
    - Kwestia różnicy między redirect a forward
- Po stronie widoku mamy tworzenie adresu:
  - $this->url(urlOptions,name,reset,encode)
    - Przykład:
      - $this->url(array('controller'=>'index','action'=>'edit','id'=>$x->id))

Obiekty request i response

- Obiekt request
  - Instancja obiektu: Zend_Controller_Request_Http
  - Dostępna w kontrolerze:
    - $this->getRequest() lub $this->_request
  - Szereg metod związanych z żądaniem:
    - getModuleName(), getControllerName(), getActionName()
    - getParams(), getParam()
    - isGet(), isPost(), isPut(), isDelete(), isHead(), isOptions()
    - getPost('var'), getQuery(), getHeader()
  - Dla większości metod getX jest odpowiednik setX
  - Więcej: http://framework.zend.com/manual/en/zend.controller.request.html

Obiekty request i response

- Obiekt response
  - Instancja obiektu
  - Dostępna w kontrolerze:
    - $this->getResponse()
    - $this->_response
  - Szereg metod związanych z odpowiedzią
    - Nagłówki: canSendHeaders() , setHeader(), setRedirect(), getHeaders(), clearHeaders(), setRawHeader(), getRawHeaders(), clearRawHeaders(), clearAllHeaders()
    - Manipulacja segmentami: setBody(), appendBody(), prepend(), append(), insert(), clearBody(), getBody()
  - Więcej: http://framework.zend.com/manual/en/zend.controller.response.html

Ciastka

- Dwie operacje:
  - Pobranie: $this->getRequest()->getCookie('foo')
  - Ustawienie:
    - $client->setCookie($cookie)
    - Bezpośrednio w PHP przez setcookie
- Do poczytania:
  - http://framework.zend.com/manual/en/zend.http.cookies.html
  - http://framework.zend.com/manual/en/zend.http.client.advanced.html

Layout, kontroler, widok

- Kontroler
  - Zawiera zbiór akcji
  - Steruje logiką aplikacji
  - Kontrolery są domyślnie w application/controllers
  - Mamy możliwość tworzenia helperów dla kontrolerów:
    - Najprościej utworzyć z poziomu Zend Studio
    - Domyślnie są w application/controllers/helpers

Layout, kontroler, widok

- Kontroler c.d.
  - Aby mieć dostępne helpery dla kontrolerów, trzeba zrobić jedną z dwóch rzeczy:
    - W index.php dopisać
      - Zend_Controller_Action_HelperBroker::addPath(APPLICATION_PATH .'/controllers/helpers');
        - Trzeba wcześniej pamiętać o załadowaniu klasy Zend_Controller_Action_HelperBroker
    - W pliku application.ini za rejestrować klasę helpera, np.:
      - resources.frontController.actionhelperpaths.PotegaHelper = APPLICATION_PATH "/controllers/helpers„

Layout, kontroler, widok

- Zbudowanie struktury opartej na layoucie sprowadza się do wykonania:
  - Tworzymy plik layoutu
    - Sugerowane foldery:
      - application/views/layouts
      - application/layouts/scripts (wskazywany przez Zend Studio)
    - W miejscach, gdzie ma się pojawić zawartość piszemy
      - $this->layout()->header [przykładowo dla header]
      - Główny skrypt widoku będzie w miejcu $this->layout()->content

Layout, kontroler, widok

- Budowa c.d.
  - Następnie musimy wskazać, że chcemy korzystać z layoutu. Mamy dwa sposoby:
    - W pliku public/index.php dopisujemy wiersz
      - Zend_Layout::startMvc(array('layoutPath' => dirname(__FILE__).'/../application/views/layouts'));
    - W pliku application/configs/application.ini dopisujemy
      - resources.layout.layoutPath = APPLICATION_PATH "/layouts/scripts" (to jest bardziej eleganckie)

Layout, kontroler, widok

- Budowa c.d.
  - Przygotowujemy składowe widoki, np. header.phtml, footer.phtml, menu.phtml
  - W metodzie init() pliku kontrolera dopisujemy dla każdego składowego widoku (przykładowo):
    - ```
      $this->_response->insert( "header",
                  $this->view->render( "header.phtml" ) );
      ```
  - I gotowe
- Uwaga: trzeba zadbać, aby w pliku index.php wczytać odpowiednie klasy, np. Zend_Layout
      ```

Layout, kontroler, widok

- Przydatnym mechanizmem są helpery do widoków
  - Tworzymy jest w folderze application/views/helpers
  - Najprościej jest tworzyć ja jako ZendFrameworkItem w ZendStudio
  - Jeśli utworzymy helper MyHelper, to wewnątrz widoku mamy metodę $this->myHelper()
    - I ZendStudio je podpowiada

Layout, kontroler, widok

- Kolejnym przydatnym mechanizmem jest tworzenie „podwidoków”
  - Zakładamy folder application/view/scripts/partials
  - Tworzymy skrypt, np. genTable
    - Wewnątrz skryptu możemy tworzyć kontrukcję $this->atrybut ($this to będzie obiekt, tablica, zmienna przekazane do podwidoku jako model)
  - Osadzamy „podwidok” za pomocą kontrukcji:
    - echo $this->partial("partials/genTable.phtml", $table);

Layout, kontroler, widok

- A jak chcemy inny layout lub w ogóle go nie chcemy?
  - Ustawienie w akcji innego layoutu:
    - $this->_helper->layout->setLayout('innylayout');
  - Wyłączenie layoutu:
    - $this->_helper->layout->disableLayout();

Layout i widoki

- Przykład
  - SimpleStart
  - SimpleStart2
- Więcej o layoutach:
  - http://framework.zend.com/manual/en/zend.layout.quickstart.html

Sesje

- Aby skorzystać z sesji, tworzymy obiekt
  - $session = new Zend_Session_Namespace() [możemy przekazać nazwę, wtedy będziemy mieć sesję nazwaną]
- Dostęp do zmiennych to po prostu
  - $session->zmienna
- Można tworzyć dowolną liczbę nazwanych sesji (i jedną domyślną)

Sesje

- Przykład
  - SessionSample

Formularze

- Klasa formularza to Zend_Form
- Możemy jej ustawić szereg standardowych parametrów
  - Action, Method, Name, itp.
- Zend_Form ma
  - kolekcję kontrolek
  - metodę valid()
  - metodę populate($dane)

Formularze

- Do budowania formularzy mamy
  - Kontrolki
  - Dekoratory
  - Walidatory
  - Filtry
- Dostępne są predefiniowane elementy z każdej grupy
  - Można jednak tworzyć własne implementacje poprzez dziedziczenie z odpowiednich klas

Formularze

- Kontrolki
  - Utworzyć można większość typów kontrolek
  - Dostępne są w przestrzeni
    - Zend_Form_Element_*
  - Przykładowe
    - Button, Captcha, Checkbox, Exception, File, Hash, Hidden, Image, MultiCheckbox, Multiselect, Password, Radio, Reset, Select, Submit, Text, TextArea

Formularze

- Dekoratory
  - Służą do określenia sposobu prezentacji
  - Można je łączyć (nakładać wiele na raz)
  - Dostępne są w przestrzeni
    - Zend_Form_Decorator_*
  - Przykładowe
    - Callback, DtDdWrapper (domyślny), Errors, Fieldset, HtmlTag, Label, Tooltip, ViewHelper, ViewScript

Formularze

- Walidatory
  - Służą do weryfikacji poprawności formularzy
  - Pozwalają także na określenie komunikatów błędów
  - Dostępne w przestrzeni
    - Zend_Validate_*
  - Przykładowe
    - Callback, Alnum, Alpha, CreditCard, Date, Digits, EmailAddress, File_*,Iban, Isbn, NotEmpty, Regex

Formularze

- Filtry
  - Pozwalają modyfikować dane wysłane z formularza
  - Dostępne w przestrzeni
    - Zend_Filter_*
  - Przykładowe
    - Boolean, Callback, Compress, Encrypt, HtmlEntities, PregReplace, StringToLower, StringToUpper, StringTrim, StripTags, Word_SeparatorToCamelCase

Formularze

- Przykłady
  - FormSample
  - FormSample2

Uwierzytelnianie i autoryzacja

- Uwierzytelnianie
  - Do uwierzytelniania jest podsystem Zend_Auth
  - „Out of the box” mamy wsparcie dla uwierzytelniania opartego o:
    - Tabele SQL
    - Digest
    - HTTP
    - LDAP
    - OpenID
  - Więcej: http://framework.zend.com/manual/en/zend.auth.html

Uwierzytelnianie i autoryzacja

- Uwierzytelnianie
  - Po wybraniu adaptera (np. opartego o tabele SQL)
    - Ustawiamy:
      $authAdapter >setIdentity($user)->setCredential($pass);
    - I robimy:
      $authResult = Zend_Auth::getInstance()->authenticate($authAdapter);
    - A następnie sprawdzamy:
      $authResult->isValid()
    - Mamy też metody:

      Zend_Auth::getInstance()->hasIdentity()
      Zend_Auth::getInstance()->clearIdentity();

Uwierzytelnianie i autoryzacja

- Uwierzytelnianie
  - Można też tworzyć własny adapter
    - Trzeba zaimplementować interfejs Zend_Auth_Adapter_Interface
    - czyli metodę authenticate
  - Więcej:
    - http://zendgeek.blogspot.com/2009/07/zend-framework-sign-up-and.html

Uwierzytelnianie i autoryzacja

- Autoryzacja jest realizowana przez Zend_Acl
- Terminologia
  - Zasób (resource)
    - Myślimy jak o kontrolerze
  - Rola (role)
    - rola chce uzyskać dostęp do zasobu
    - można określić operacje, które rola może wykonać na zasobie
      - myślimy o nich jak o akcjach kontrolera

Uwierzytelnianie i autoryzacja

- Zasób
  - Implementuje interfejs Zend_Acl_Resource_Interface
    - Jest tylko jedna metoda: getResourceId()
    - Domyślna implementacja: Zend_Acl_Resource
  - Dany zasób **może** mieć *parenta* (**dokładnie jednego**)
    - czyli zasoby można organizować w drzewa, co pozwala na dziedziczenie uprawnień
  - Zend_Acl ma także mechanizm uprawnień na zasobach, np. create, read, update, delete
  - O zasobie możemy myśleć w kontekście kontrolera

Uwierzytelnianie i autoryzacja

- Rola
  - Implementuje interfejs Zend_Acl_Role_Interface
    - Jest tylko jedna metoda: getRoleId()
    - Domyślna implementacja: Zend_Acl_Role
  - Dana rola może dziedzić jednej lub wielu ról
    - czyli ponownie mamy drzewo, które pozwala na dziedziczenie uprawnień
    - w przypadku konfliktów są reguły ich rozwiązywania
    - przykład wielodziedziczenia i konfliktów: http://framework.zend.com/manual/en/zend.acl.introduction.html

Uwierzytelnianie i autoryzacja

- Utworzenie Access Control List
  - Najpierw tworzymy obiekt
    - $acl = new Zend_Acl();
  - Rejestrujemy role, ewentualnie wskazując parentów
    - $parents = array('guest', 'member');
      $acl->addRole(new Zend_Acl_Role('admin'), $parents);
  - Rejestracja zasobu
    - $acl->add(new Zend_Acl_Resource('articles');
  - Ustalamy prawa
    - $acl->deny('guest', 'articles');
    - $acl->allow('member', 'articles');
  - Na końcu możemy odpytać
    - echo $acl->isAllowed('member', 'articles') ? 'allowed' :'denied';

Uwierzytelnianie i autoryzacja

- Ostatecznie
  - tworzymy plugin Acl (np. Plugin_Acl)
    - Dziedziczy po klasie Zend_Controller_Plugin_Abstract
  - W application/configs/application.ini dodajemy
    - resources.frontController.plugins.acl = "Plugin_Acl"
  - Na końcu w index.php dodajemy
    - Zend_Controller_Front::getInstance()
      ->registerPlugin(new Plugin_Acl())

Uwierzytelnianie i autoryzacja

- Autoryzacja
  - Więcej:
    - http://framework.zend.com/manual/en/zend.acl.html
    - http://zendguru.wordpress.com/2008/11/05/zend-framework-acl-with-example/
    - http://devzone.zend.com/article/1665

Uwierzytelnianie i autoryzacja

- Przykład
  - Auth