# Aplikacje WWW - laboratorium

## *CakePHP*

Celem ćwiczenia jest przygotowanie prostej aplikacji internetowej wykorzystującej architekturę szkieletową CakePHP. Ćwiczenie prezentuje podstawowe aspekty tworzenia aplikacji w oparciu o architekturę szkieletową (ang. *application framework*) bazującą na paradygmacie Model-View-Controller (MVC). W ramach tutorialu powstanie prosta aplikacja zawierająca przykładowy kontroler, obiekty modelu, obiekty widoku oraz obiekty pomocnicze. Zaprezentowane także zostaną przykłady wykorzystania funkcjonalności wbudowanej w architekturę szkieletową (walidacja danych, odwzorowanie obiektowo-relacyjne, itp.)

Ćwiczenie można wykonać na dowolnym komputerze, którym zainstalowano serwer HTTP (np. Apache) z obsługą PHP oraz bazę danych MySQL. Rozwiązania ćwiczeń omawianych w poniższym zestawie zostały przygotowane z wykorzystaniem pakietu XAMPP.

**UWAGA**: CakePHP wykorzystuje bardzo wiele konwencji związanych z właściwym nazewnictwem elementów (kontrolerów, modeli, widoków). Wybór poszczególnych nazw dla klas i plików **jest nieprzypadkowy**, w celu płynnego wykonania tutorialu należy kierować się ściśle podanymi zaleceniami dotyczącymi nazw obiektów.

1. Pobierz ze strony https://github.com/cakephp/cakephp/tags wersję 2.9.8 architektury CakePHP i rozpakuj archiwum do katalogu `xampp/htdocs/cake`.
2. Uruchom program XAMPP (serwer Apache i bazę danych MySQL), a następnie uruchom program phpMyAdmin (przycisk Admin na panelu kontrolnym programu XAMPP). Utwórz bazę danych o nazwie `cakephp`. Następnie, utwórz w bazie danych `cakephp` tabelę `books` o następującym schemacie (upewnij się, że łańcuchy znaków będą kodowane w utf8_polish_ci, pole Collation)

| kolumna | typ i własności |
|---|---|
| **id** | INT UNSIGNED AUTO_INCREMENT PRIMARY KEY |
| **title** | VARCHAR(50) |
| **author** | VARCHAR(20) |
| **genre** | VARCHAR(20) |

   Wprowadź do nowoutworzonej tabeli kilka przykładowych rekordów

3. Przejdź do katalogu `/app/Config/` i utwórz kopię pliku `database.php.default`. Nazwij nowy plik `database.php`. Edytuj utworzony plik i ustaw w zmiennej `$default` parametry logowania się do bazy danych

```
var $default = array(
      'datasource' => 'Database/Mysql',
      'persistent' => false,
      'host' => 'localhost',
      'login' => 'root',
      'password' => '',
      'database' => 'cakephp',
      'prefix' => '',
      'encoding' => 'utf8',
);
```

4. Przejdź do strony http://localhost/cake/ i upewnij się, że lokalna instalacja się powiodła.
5. Pierwszym krokiem ćwiczenia będzie utworzenie klasy modelu, która będzie reprezentowała tabelę w bazie danych. Instancje klasy modelu będą odpowiadały wierszom w bazie danych. Przejdź do katalogu `/app/Model/` i utwórz w nim plik `Book.php`, a następnie umieść w pliku poniższy kod.

```
<?php
  class Book extends AppModel {
    var $name = 'Book';
  }
?>
```

6. W kolejnym kroku utworzysz najprostszy kontroler i wyposażysz go w metodę `index()`, która zostanie automatycznie wywołana w odpowiedzi na żądanie HTTP. Kontroler dla modelu `Book` powinien nosić nazwę `BooksController` i być umieszczony w pliku `BooksController.php`, w katalogu `/app/Controller`. Przejdź do tego katalogu i utwórz wspomniany plik, wypełniając go poniższą treścią (akcja `set()` służy do przekazania danych z kontrolera do widoku).

```
<?php
  class BooksController extends AppController {
    var $name = 'Books';

    var $helpers = array('Html','Form');

    function index() {
      $this->set('books', $this->Book->find('all'));
    }
  }
?>
```

7. Przejdź do strony http://localhost/cake/books/index i sprawdź, czy strona działa prawidłowo.

8. Przejdź do katalogu `/app/View` i utwórz nowy katalog `Books`. Wejdź do nowego katalogu i utwórz w nim szablon widoku o nazwie odpowiadającej nazwie akcji, która aktywizuje dany widok (`index.ctp`). Umieść w pliku widoku poniższy kod:

```
<h1>Books</h1>

<table>
<tr>
  <th>Id</th>
  <th>Title</th>
  <th>Author</th>
  <th>Genre</th>
  <th></th>
</tr>

<?php foreach ($books as $book): ?>
  <tr>
    <td><?php echo $book['Book']['id']; ?></td>
    <td>
      <?php echo $this->Html->link($book['Book']['title'],
            array('action' => 'view', $book['Book']['id']));?>
    </td>
    <td>
      <?php echo $book['Book']['author']; ?>
    </td>
    <td>
      <?php echo $book['Book']['genre']; ?>
    </td>
    <td>
      <?php echo $this->Html->link('Usuń',
            array('action' => 'delete', $book['Book']['id']),
            null, 'Czy jesteś pewna(y)?')?>
      <?php echo $this->Html->link('Edytuj',
            array('action' => 'edit', $book['Book']['id']));?>
    </td>
  </tr>
  <?php endforeach; ?>
</table>
```

Przeanalizuj starannie powyższy przykład. Zobacz, w jaki sposób następuje przejęcie danych wysłanych przez kontroler. Zauważ różne sposoby wywołania metody `Html→link()` (Html to komponent pomocniczy rejestrowany automatycznie w każdej aplikacji). Uruchom aplikację i przejdź do adresu http://localhost/cake/books/

9. Akcje zdefiniowane w poprzednim widoku (`view, delete, edit`) nie zostały jeszcze zdefiniowane. Przejdź do kontrolera i dodaj do definicji klasy poniższą metodę

```
function view($id) {
  $this->Book->id = $id;
  $this->set('book', $this->Book->read());
}
```

10. Przejdź do katalogu `/app/View/Books` i utwórz nowy plik o nazwie `view.ctp`, a następnie umieść w nim poniższy kod, uruchom aplikację i przetestuj jego działanie.

```
<ul>
  <li><b>tytuł</b>:<?php echo $book['Book']['title']?></li>
  <li><b>autor</b>:<?php echo $book['Book']['author']?></li>
  <li><b>gatunek</b>:<?php echo $book['Book']['genre']?></li>
</ul>
<a href="/cake/books">powrót</a>
```

11. W następnym kroku obsłużysz procedurę dodawania nowych krotek. Wróć do definicji kontrolera i umieść tam następującą funkcję:

```
function add() {
  if (!empty($this->data)) {
    if($this->Book->save($this->data)) {
      $this->Flash->set('Książka zosała dodana');
      $this->redirect(array('action'=>'index'));
    }
  }
}
```

12. Po zaimplementowaniu logiki dodawania nowych książek możesz przejść do widoku udostępniającego formatkę do dodawania książek. Utwórz w katalogu `/app/View/Books` plik `add.ctp` i umieść w nim:

```
<h1>Dodaj książkę</h1>
<?php
  $options = array('dramat' =>'dramat', 'komedia' =>'komedia');

  echo $this->Form->create('Book');
  echo $this->Form->input('title');
  echo $this->Form->input('author');
  echo $this->Form->input('genre',
              array('options'=>$options, 'default'=>'dramat'));
  echo $this->Form->end('Zapisz');
?>
```

13. Wróć do pliku `index.ctp` i przed znacznikiem `<table>` umieść link umożliwiający przejście do formularza dodawania nowej książki.

```
<p>
  <?php echo $this->Html->link('Dodaj',
                          array('action' => 'add')); ?>
</p>
```

14. CakePHP zawiera bardzo bogaty silnik walidacji danych. Przejdź do edycji źródła klasy modelu i wprowadź zmienną `$validate`, która zawiera reguły walidacji. Następnie, sprawdź, czy możesz dodać nowy rekord nie podając tytułu wydarzenia.

```
var $validate = array('title' => array('rule' => 'notBlank'));
```

15. Następnym krokiem jest oprogramowanie akcji usuwania książek. Przejdź do kontrolera i dodaj metodę, która będzie wywoływana w momencie kliknięcia na link Usuń.

```
function delete($id) {
  if ($this->Book->delete($id)) {
    $this->Flash->set('Książka została usunięta');
    $this->redirect(array('action' => 'index'));
  }
}
```

16. Ostatnią operacją CRUD która nie została jeszcze oprogramowana jest operacja edycji. Przejdź do katalogu `app/View/Books` i utwórz plik `edit.ctp` z następującą treścią:

```
<h1>Edytuj książkę</h1>
<?php
  $options = array('dramat' => 'dramat','komedia' => 'komedia');

  echo $this->Form->create('Book', array('url' => 'edit'));
  echo $this->Form->input('id', array('type' => 'hidden'));
  echo $this->Form->input('title');
  echo $this->Form->input('author');
  echo $this->Form->input('genre',
              array('options'=>$options, 'default'=>'dramat'));
  echo $this->Form->end('Zapisz');
?>
```

17. Edytuj kontroler aplikacji i dodaj logikę biznesową odpowiedzialną za edycję książek.

```
function edit($id = null) {
  $this->Book->id = $id;
  if (empty($this->data)) {
    $this->data = $this->Book->read();
  } else {
    if ($this->Book->save($this->data)) {
      $this->Flash->set('Książka została zmieniona');
      $this->redirect(array('action' => 'index'));
    }
  }
}
```

18. CakePHP wykorzystuje mechanizm routingu do odwzorowania żądań http na akcje konkretnych kontrolerów. Głównym plikiem konfiguracyjnym jest `/app/Config/routes.php`. Edytuj ten plik w celu dodania nowej reguły, która przekieruje żądania do korzenia aplikacji do metody `index()` kontrolera `BooksController`. Wykomentuj domyślną regułę (dotyczącą kontrolera `PagesController` dostarczanego domyślnie ze środowiskiem) i zamień ją na poniższą. Następnie otwórz w przeglądarce adres http://localhost/cake

```
Router::connect('/',
  array('controller' => 'books', 'action' => 'index'));
```

19. CakePHP dostarcza także automatycznego mechanizmu tworzenia całego szkieletu CRUD dla obiektów w bazie danych (kontroler, widoki, modele). Aby wykorzystać tę własność środowiska, uruchom aplikację PHPMyAdmin i utwórz w bazie danych następujące tabele:

```
CREATE TABLE users (
    id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password CHAR(40) NOT NULL,
    group_id INT(11) NOT NULL,
    created DATETIME,
    modified DATETIME
);


CREATE TABLE groups (
    id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    created DATETIME,
    modified DATETIME
);
```

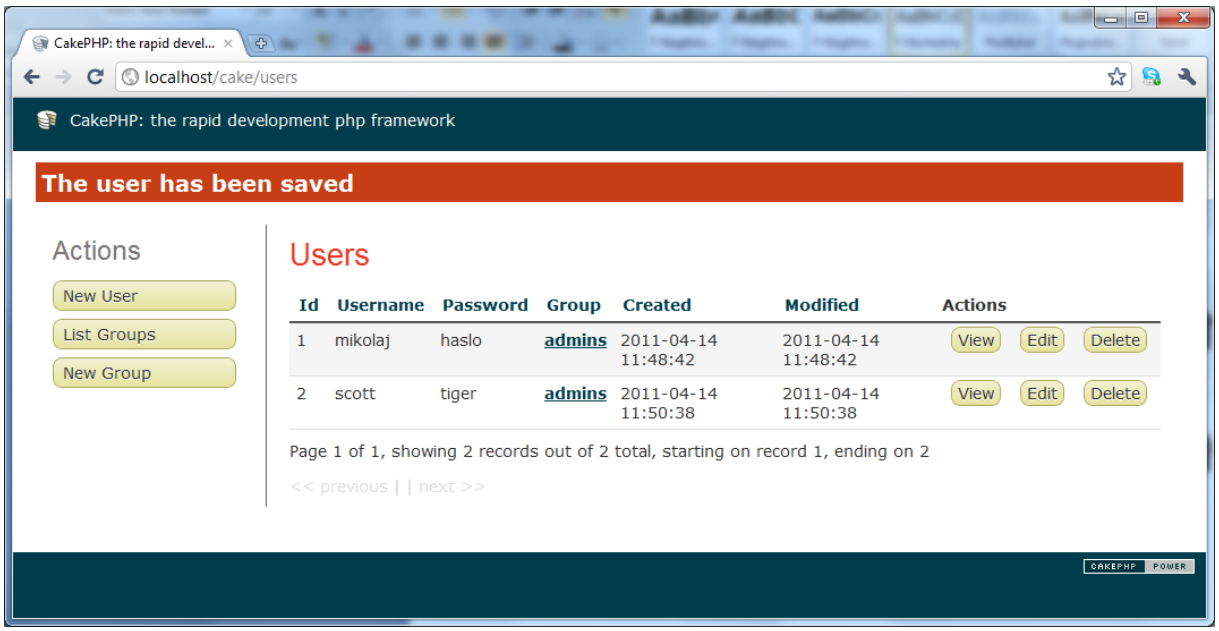
20. Upewnij się, że foldery `xampp/php` i `cake/app/console` znajdują się na ścieżce. Jeśli nie, dodaj je do zmiennej `PATH` (upewnij się, że wpisujesz właściwe katalogi)

```
C:\> set PATH=%PATH%;c:\xampp\php;c:\xampp\htdocs\cake\app\console
```

21. Przejdź do katalogu `cake/app` i uruchom polecenie do automatycznej generacji szkieletu aplikacji. Wybierz najpierw tabelę `GROUPS` i wygeneruj właściwe pliki, następnie powtórz to samo dla tabeli `USERS`.

```
C:\xampp\htdocs\cake\app> cake bake all
```

22. Otwórz w przeglądarce adres http://localhost/cake/users, dodaj nową grupę, a następnie dodaj dwóch użytkowników. Zapoznaj się z wygenerowanymi plikami kontrolerów, modeli i widoków.



23. Wykorzystaj poniższy kod do utworzenia tabeli `PRACOWNICY` i wypełnienia jej danymi. Utwórz szkielet aplikacji CRUD do obsługi tabeli `PRACOWNICY`. Popraw wygenerowane automatycznie pliki w następujący sposób:
    - dodaj do modelu informację o tym, że nazwiska i etaty pracowników są wymagane
    - dodaj walidację płacy podstawowej, musi się zawierać w przedziale <0-2000>
    - pole do wprowadzania etatu powinno być statyczną listą rozwijaną

```
DROP TABLE IF EXISTS `employees`;
CREATE TABLE `employees` (
  `id` int AUTO_INCREMENT NOT NULL,
  `nazwisko` varchar(15) default NULL,
  `imie` varchar(15) default NULL,
  `etat` varchar(10) default NULL,
  `id_szefa` decimal(4,0) default NULL,
  `zatrudniony` date default NULL,
  `placa_pod` decimal(6,2) default NULL,
  `placa_dod` decimal(6,2) default NULL,
  `id_zesp` decimal(2,0) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM;

INSERT INTO `employees` (`id`, `nazwisko`, `imie`, `etat`, `id_szefa`,
`zatrudniony`, `placa_pod`, `placa_dod`, `id_zesp`)
VALUES
(100, 'Marecki', 'Jan', 'DYREKTOR', NULL, '1968-01-01', 4730.00, 980.50, 10),
(110, 'Janicki', 'Karol', 'PROFESOR', 100, '1973-05-01', 3350.00, 610.00, 40),
(120, 'Nowicki', 'Pawel', 'PROFESOR', 100, '1977-09-01', 3070.00, NULL, 30),
(130, 'Nowak', 'Piotr', 'PROFESOR', 100, '1968-07-01', 3960.00, NULL, 20),
(140, 'Kowalski', 'Krzysztof', 'PROFESOR', 130, '1975-09-15', 3230.00, 805.00, 20),
(150, 'Grzybowska', 'Maria', 'ADIUNKT', 130, '1977-09-01', 2845.50, NULL, 20),
(160, 'Krakowska', 'Joanna', 'SEKRETARKA', 130, '1985-03-01', 1590.00, NULL, 20),
(170, 'Opolski', 'Roman', 'ASYSTENT', 130, '1992-10-01', 1839.70, 480.50, 20),
(190, 'Kotarski', 'Konrad', 'ASYSTENT', 140, '1993-09-01', 1971.00, NULL, 20),
(180, 'Makowski', 'Marek', 'ADIUNKT', 100, '1985-02-20', 2610.20, NULL, 10),
(200, 'Przywarek', 'Leon', 'DOKTORANT', 140, '1994-07-15', 900.00, NULL, 30),
(210, 'Kotlarczyk', 'Stefan', 'DOKTORANT', 130, '1993-10-15', 900.00, 570.60, 30),
(220, 'Siekierski', 'Mateusz', 'ASYSTENT', 110, '1993-10-01', 1889.00, NULL, 20),
(230, 'Dolny', 'Tomasz', 'ASYSTENT', 120, '1992-09-01', 1850.00, 390.00, NULL);
```