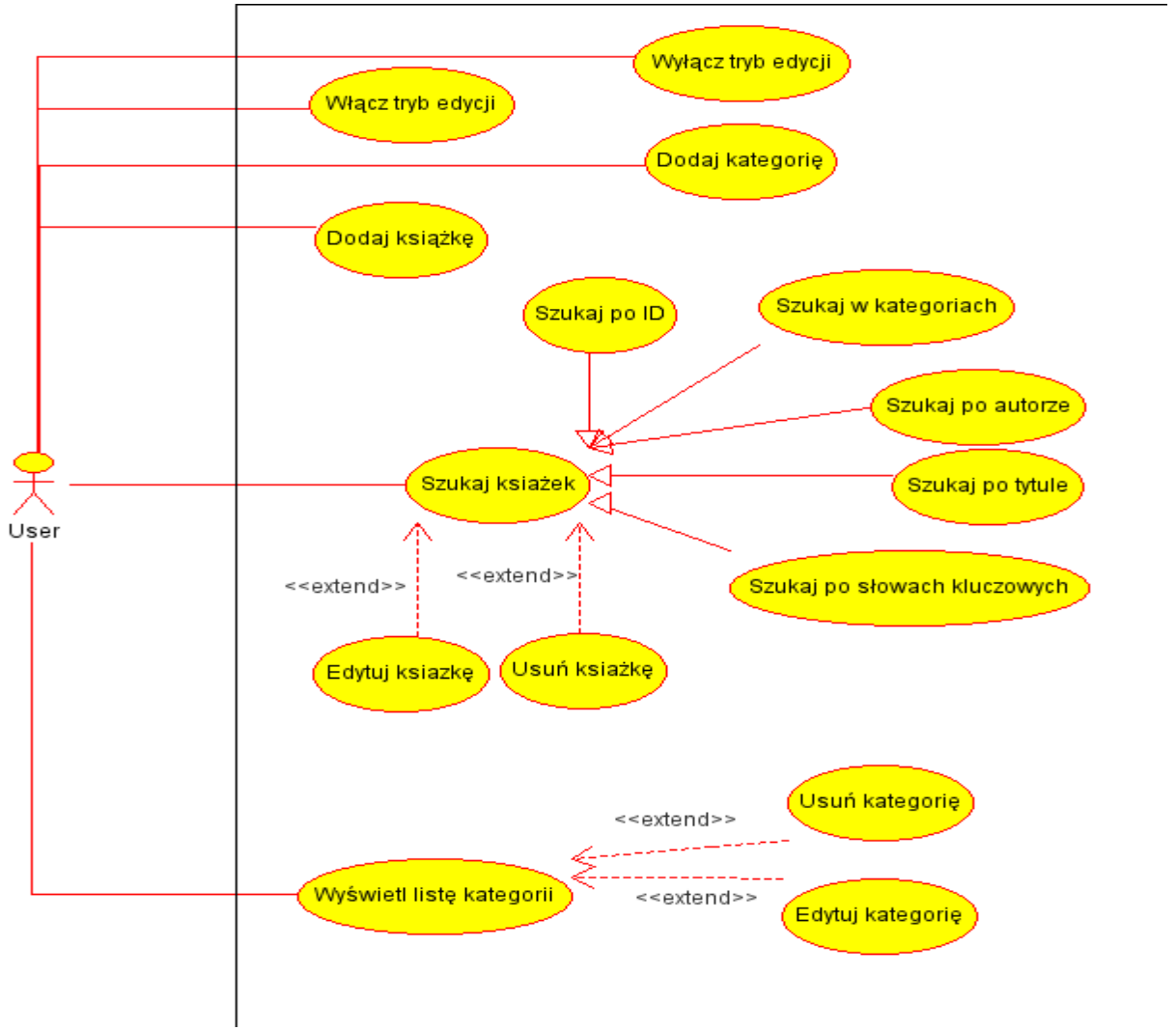
Podstawy programowania III

WYKŁAD 6

Jan Kazimirski

Projekt: „Katalog książek elektronicznych” c.d.

Diagram przypadków użycia



Iteracja 1

- Zaprojektowanie panelu głównego aplikacji
- Realizacja przypadków użycia:
  - Włącz tryb edycji
  - Wyłącz tryb edycji

Panel główny

Obszar lewy:

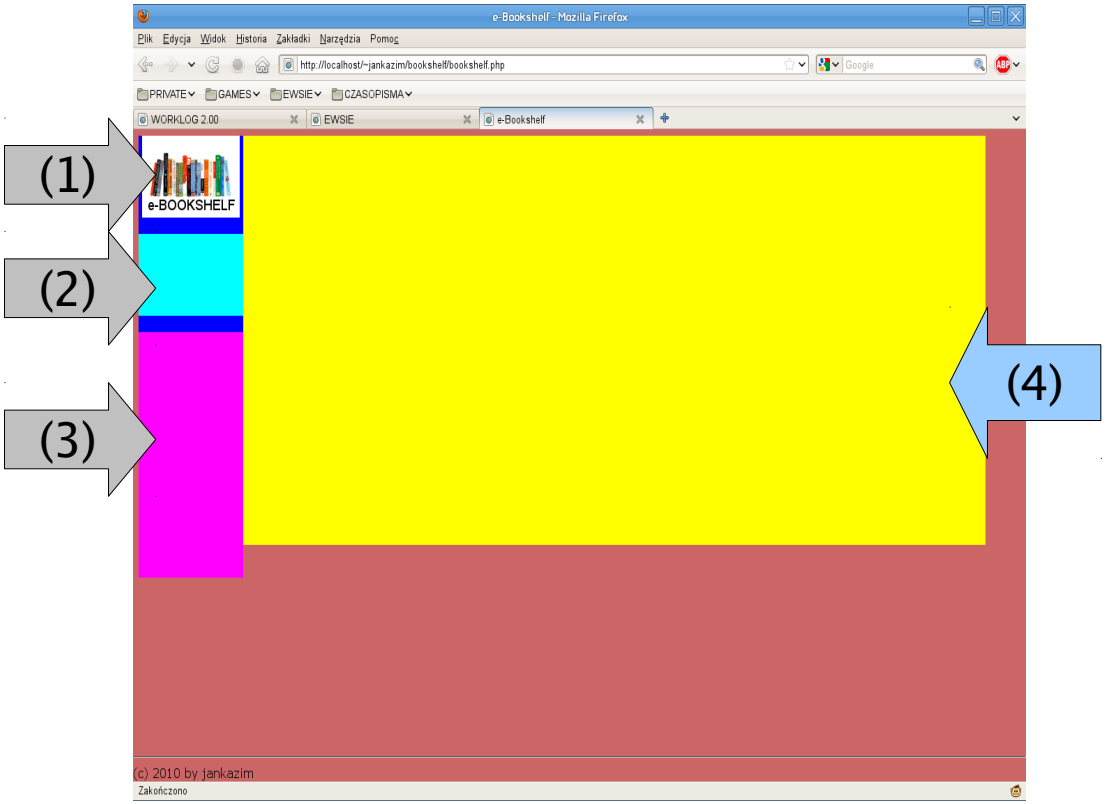
logo aplikacji (1)

przycisk zmiany trybu (2)

menu główne (3)

Obszar prawy:

główny panel roboczy (4)



Klasa HtmlTools

Metody

head – wyświetlanie nagłówka strony

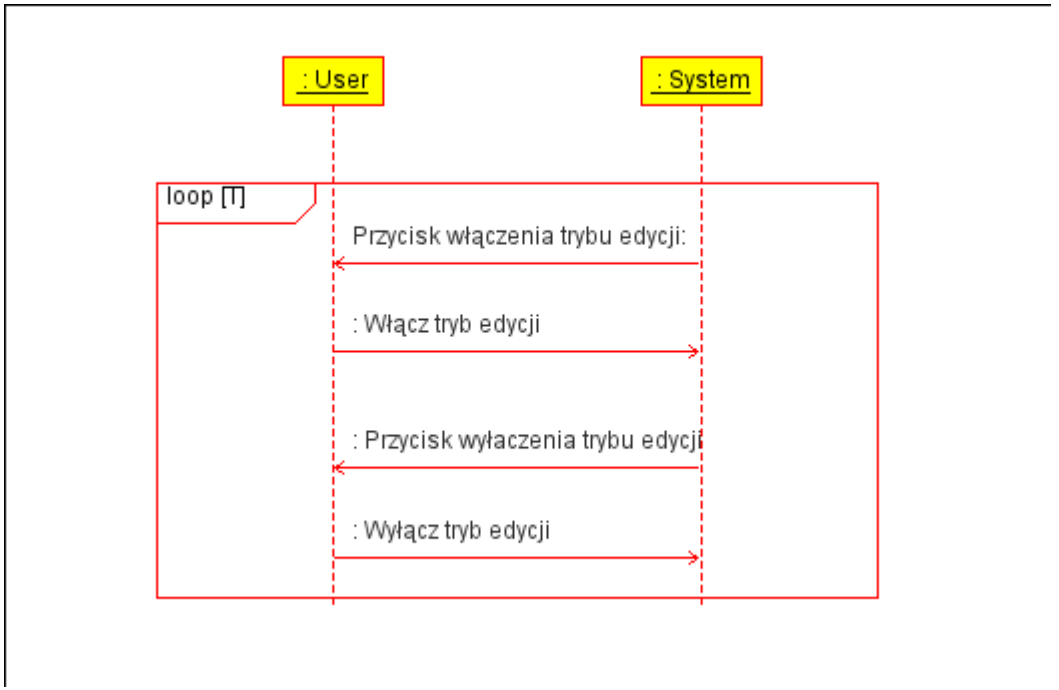
tail – wyświetlanie stopki strony

page – wyświetlanie układu strony (lpan – część dynamiczna lewego panelu, rpan – prawy panel)

HtmlTools

```php
<?php

class HtmlTools {
  public function head() {
    echo '<html><head>'."\n";
    echo '<TITLE>e-Bookshelf</TITLE>'."\n";
    echo '<meta http-equiv="Content-Type" content="text/html; charset=utf-8">'."\n";
    echo '<link rel="Stylesheet" type="text/css" href="bookshelf.css" />'."\n";
    echo '</head><body>';
  }

  public function tail() {
    echo '<div id="tail"><hr>(c) 2010 by jankazim</div>'."\n";
    echo '</body></html>'."\n";
  }

  public function page($lm,$rs) {
    self::head();

    // left side (top,middle,bottom)
    echo '<div id="lside">'."\n";
    echo '<div id="ltop"><img src="img/books.png"></div>'."\n";
    echo '<div id="lmid">'.$lm.'</div>'."\n";
    echo '<div id="lbot">'.$menu.'</div>'."\n";
    echo '</div>';

    // right side (top,bottom)
    echo '<div id="rside">'.$rs.'</div>'."\n";

    self::tail();
  }
};
```

Zmiana trybu edycji

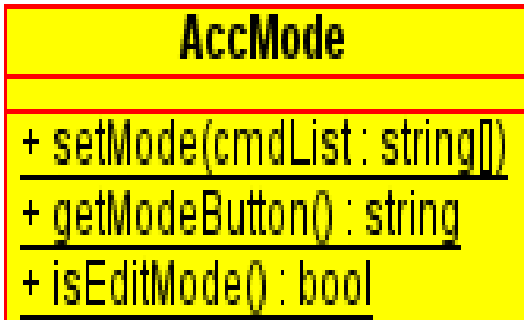


Zmiana trybu pracy: tryb edycji wyłączony lub włączony

Klasa AccMode

Odpowiedzialność:

1. Wyświetlenie przycisków włączenia i wyłączenia trybu edycji
2. Informacja o aktualnym trybie
3. Ustawienie domyślnego trybu edycji.



Dygresja - TDD

- Podejście klasyczne:
  projektowanie -> implementowanie -> testowanie
- TDD – Test Driven Development:
  projektowanie -> tworzenie testów ->
  (testowanie -> implementowanie -> testowanie)..
- W metodologii TDD najpierw tworzymy testy a później implementujemy funkcjonalność.

Dygresja – TDD c.d.

- Typowy przebieg czynności
  - Projektujemy klasę/moduł i tworzymy pusty szkielet
  - Tworzymy testy dla poszczególnych metod i funkcji
  - Uruchamiamy testy – wszystkie zakończą się niepowodzeniem
  - Implementujemy fragment funkcjonalności
  - Uruchamiamy testy – część testów powinna się powieść
  - Dodajemy funkcjonalność, refaktoryzujemy itp.
  - Uruchamiamy testy ...
  - Kod jest gotowy gdy wszystkie testy są poprawne

Dygresja – TDD c.d.

- Zalety metodologii TDD
  - Lepszy projekt (projektowanie testu „wymusza” lepszy design metod).
  - Mniej błędów w kodzie.
  - Bezpieczniejsza manipulacja kodem (refaktoryzacja, zmiany koncepcji).
  - Automatyzacja procesu testowania.
  - Lepsza kontrola nad postępem prac nad projektem (liczba poprawnych vs. niepoprawnych testów).

TDD - xUnit

- Środowisko xUnit – tworzenie i uruchamianie testów jednostkowych.
- Dostępne dla wielu języków programowania m.in.:
  - Java (Junit)
  - C++ (CppUnit)
  - PHP (PHPUnit)

AccMode – szkielet klasy

```php
<?php

// Access mode manipulations.
class AccMode {

  // Set default and change current mode.
  public function setMode($cmdList) {}

  // Get HTML code for the change mode button.
  public function getModeButton() {}

  // Return TRUE if edit mode is on.
  public function isEditMode() {}

};

?>
```

AccModeTest – testy jednostkowe

```
$_SESSION = array();

require_once("AccMode.php");

// Unit tests for AccMode class
class AccModeTest extends PHPUnit_Framework_TestCase {

 public function testDefaultValue() {
  AccMode::setMode(array());
  $this->assertEquals("View",$_SESSION["AccMode"]);
 }

 public function testChangeEditMode() {
  AccMode::setMode(array("cmd"=>"E+"));
  $this->assertEquals("Edit",$_SESSION["AccMode"]);
  AccMode::setMode(array("cmd"=>"E-"));
  $this->assertEquals("View",$_SESSION["AccMode"]);
 }

 public function testRaportEditMode() {
  AccMode::setMode(array("cmd"=>"E+"));
  $this->assertEquals(TRUE,AccMode::isEditMode());
  AccMode::setMode(array("cmd"=>"E-"));
  $this->assertEquals(FALSE,AccMode::isEditMode());
 }
};
```

Iteracja 1.0 - testy

```
PHPUnit 3.5.7 by Sebastian Bergmann.

EEF

Time: 0 seconds, Memory: 4.50Mb

There were 2 errors:

1) AccModeTest::testDefaultValue
Undefined index: AccMode

/home/jankazim/public_html/bookshelf/AccModeTest.php:12

2) AccModeTest::testChangeEditMode
Undefined index: AccMode

/home/jankazim/public_html/bookshelf/AccModeTest.php:17

--
```

Dwa błędy, jeden wynik nieprawidłowy

BŁĄD

BŁĄD

Iteracja 1.0 – testy c.d.

There was 1 failure:

1) AccModeTest::testRaportEditMode

Failed asserting that <null> matches expected <boolean:true>.

/home/jankazim/public_html/bookshelf/AccModeTest.php:24

Test zakończony niepowodzeniem

FAILURES!

Tests: 3, Assertions: 1, Failures: 1, Errors: 2.

Podsumowanie

Iteracja 1.1 – AccMode::setMode

```
class AccMode {

 // Set default and change current mode.
 public function setMode($cmdList) {
   if(!isset($_SESSION['AccMode']))  $_SESSION["AccMode"]="View";
   if(isset($cmdList["cmd"])) {
     if($cmdList["cmd"]=="E+") $_SESSION["AccMode"]="Edit";
     if($cmdList["cmd"]=="E-") $_SESSION["AccMode"]="View";
   };
 }

.....

};
```

```
..F

There was 1 failure:

1) AccModeTest::testRaportEditMode
Failed asserting that <null> matches expected <boolean:true>.

FAILURES!
Tests: 3, Assertions: 4, Failures: 1.
```

Iteracja 1.2 – AccMode::isEditMode

```
class AccMode {

...

  // Return TRUE if edit mode is on.
  public function isEditMode() {
    return ($_SESSION['AccMode']=="Edit");
  }

};
```

```
PHPUnit 3.5.7 by Sebastian Bergmann.

...

Time: 0 seconds, Memory: 4.25Mb

OK (3 tests, 5 assertions)
```

Iteracja 1.3 – AccMode::getModeButton

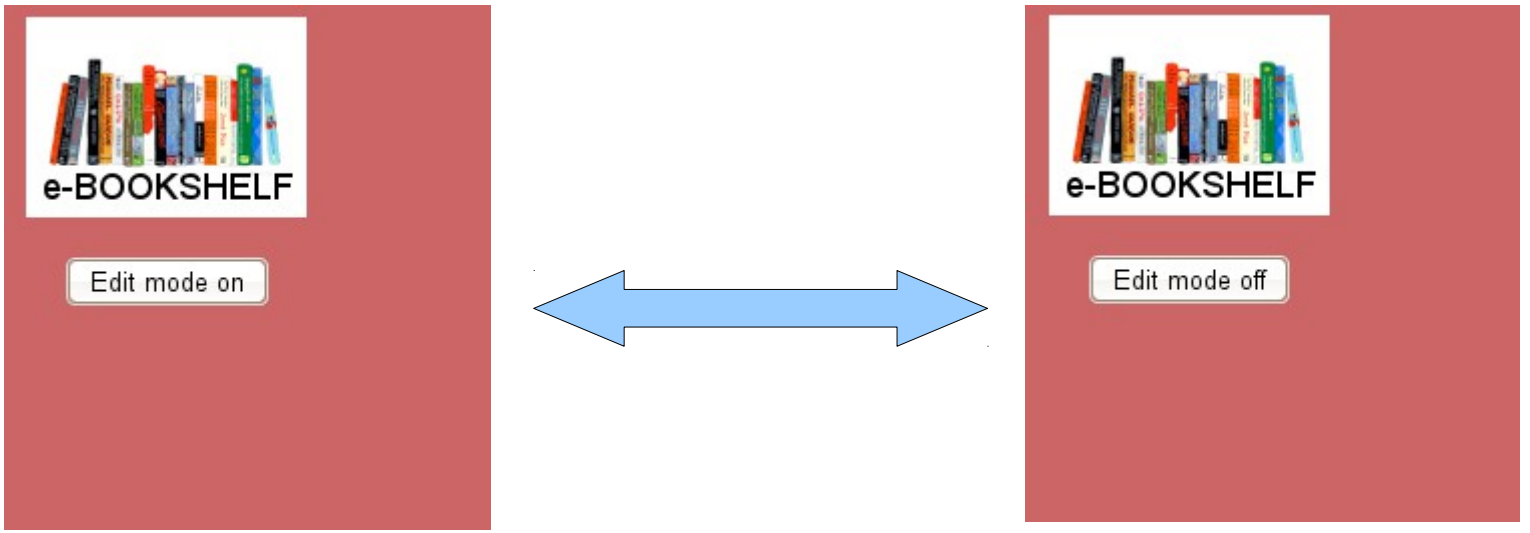
```
class AccMode {

...

 // Get HTML code for the change mode button.
 public function getModeButton() {
   $str = "<form action=".$_SERVER['PHP_SELF']." method=\"GET\">";
   if($_SESSION['AccMode']=="Edit") {
     $str.= "<input type=\"hidden\" name=\"cmd\" value=\"E-\">";
     $str.= "<input type=\"submit\" value=\"Edit mode on\"></form>";
   } else {
     $str.= "<input type=\"hidden\" name=\"cmd\" value=\"E+\">";
     $str.= "<input type=\"submit\" value=\"Edit mode off\"></form>";
   };
   return $str;
 }

...

};
```

OSTROŻNIE!!!
Brak pokrycia testami

Iteracja 1.4 – program główny

```php
<?php

session_start();
date_default_timezone_set('UTC');
error_reporting(E_ALL);

require_once("HtmlTools.php");
require_once("AccMode.php");

AccMode::setMode($_REQUEST);
$rpan = AccMode::getModeButton();

Html::page($rpan,"");

?>
```

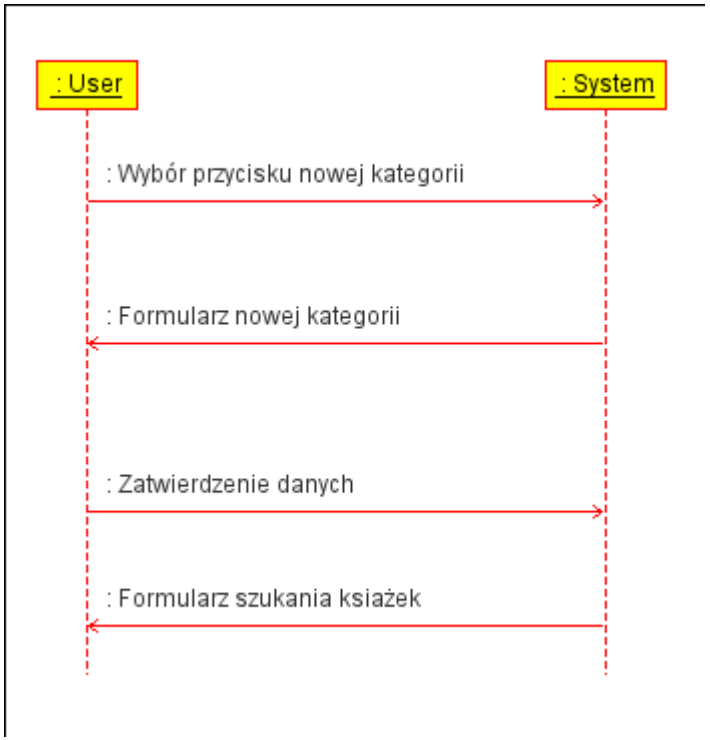
Iteracja 1 - KONIEC



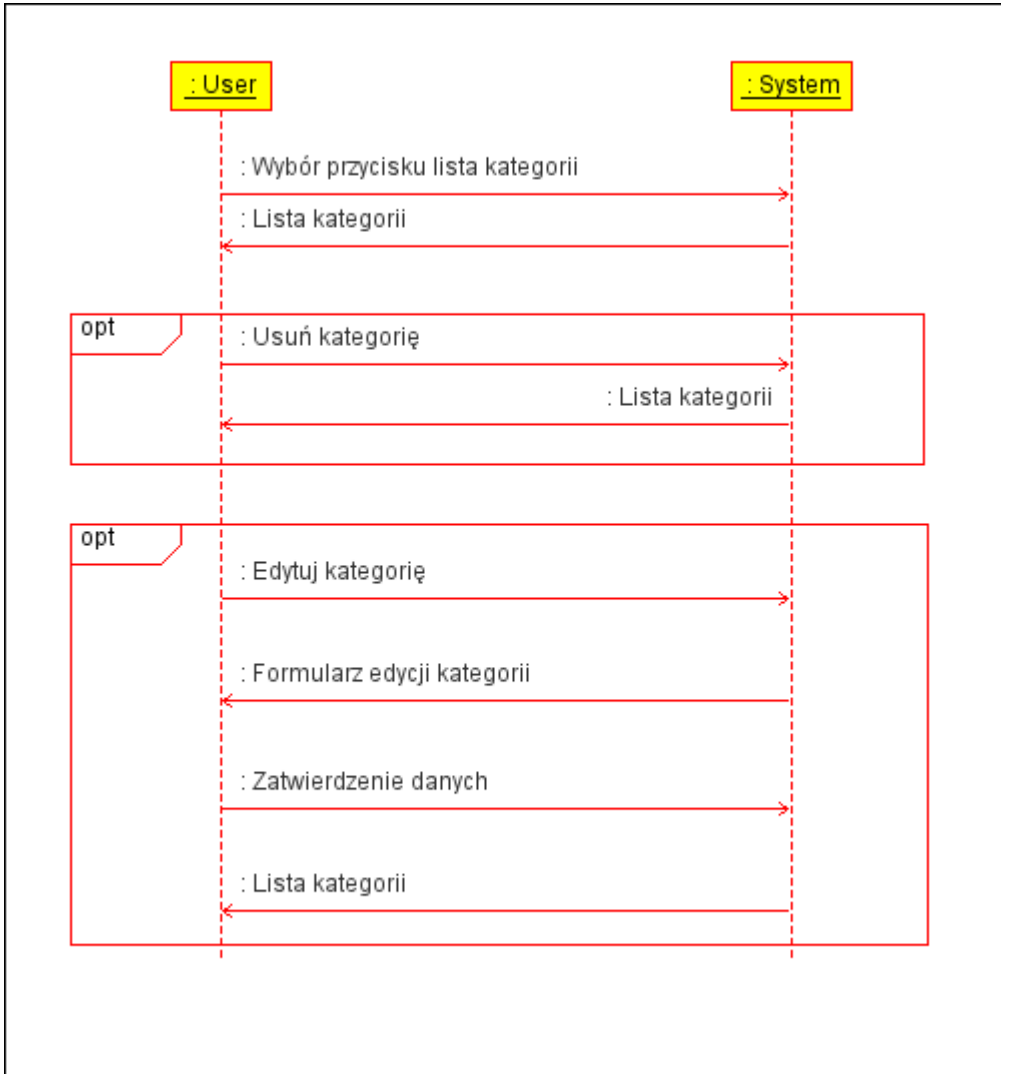
Iteracja 2

- Realizacja przypadków użycia
  - Dodaj kategorię
  - Wyświetl listę kategorii
  - Edytuj kategorię
  - Usuń kategorię

Obsługa kategorii

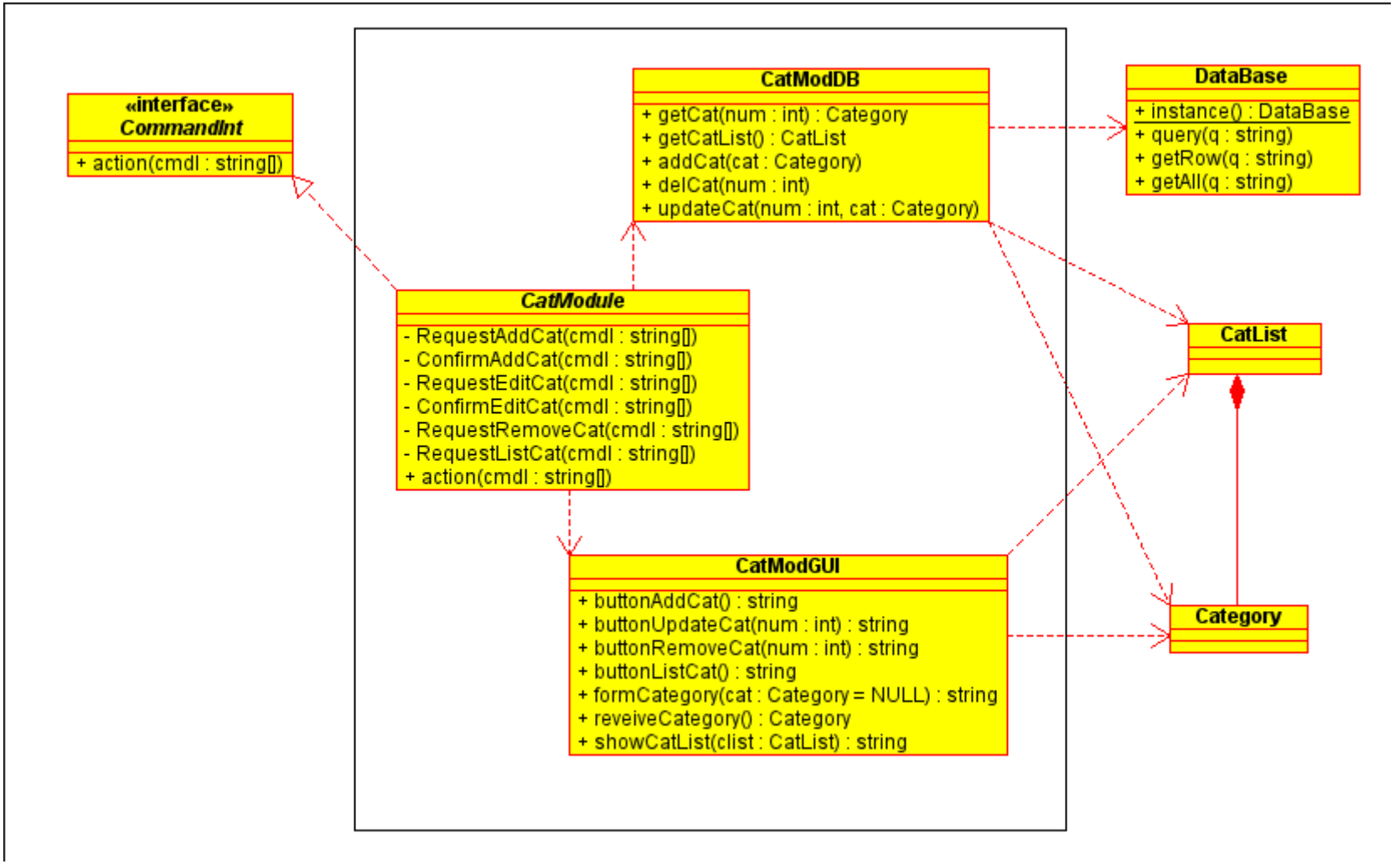


Dodanie kategorii



Lista kategorii, dodaj, usuń kategorię

Moduł CatModule - klasy



Klasa DataBase

Klasa DataBase – realizacja wzorca singleton.

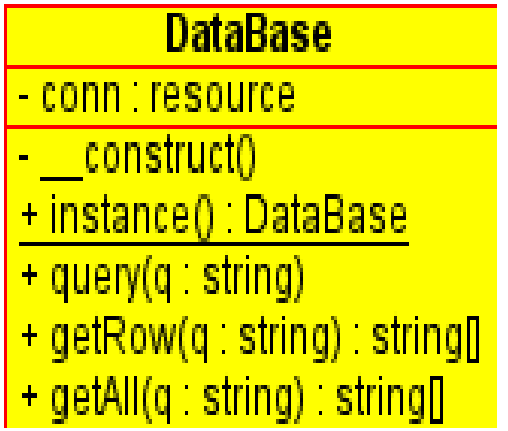
Realizacja zapytań do bazy danych:

query – zapytanie bez danych zwrotnych

getRow – zapytanie z jednym rezultatem

getAll – zapytanie zwracające listę rezultatów

Rezultaty zwracane w postaci tablicy 1 lub 2-wymiarowej zmiennych typu string.



DataBase - szkielet

```php
<?php

// PostgreSQL database interface
class DataBase {

  private $conn = NULL;

  private function __constructor() {}

  // Return DataBase object
  public function instance() {}

  // Execute query with no results
  public function query($q) {}

  // Execute query with one result
  public function getRow($q) {}

  // Execute query with many results
  public function getAll($q) {}
```

DataBase – testy jednostkowe

- Zaimplementowane testy:
  - testConnection
  - testValidQuery
  - testInvalidQuery
  - testValidGetRow
  - testInvalidGetRow
  - testValidGetAll
  - testInvalidGetAll

Na tym etapie testy nie dają się nawet uruchomić:

FPHP Fatal error: Call to a member function query() on a non-object in /home/jankazim/public_html/bookshelf/DataBaseTest.php on line 15

DataBase – implementacja (1)

```
  private $conn = NULL;

  private function __construct() {
    $this->conn = @pg_connect('host=localhost dbname=jankazim'.
                              ' user=jankazim password=jankazim');
    if($this->conn==NULL)throw new Exception("DatabaseConnectionError");
  }

  // Return DataBase object
  public function instance() {
    static $objDB;
    if(!isset($objDB)) $objDB = new DataBase();
    return $objDB;
  }

...
};
```

testConnection

testValidQuery

testInvalidQuery

testValidGetRow

testInvalidGetRow

testValidGetAll

testInvalidGetAll

DataBase – implementacja (2)

```
class DataBase {

...

 // Execute query with no results
 public function query($q) {
   $pgq=@pg_query($this->conn,$q);
   if($pgq==NULL) throw new Exception("DataBaseInvalidQueryError");
   return TRUE;
 }

...

};
```

testConnection

testValidQuery

testInvalidQuery

testValidGetRow

testInvalidGetRow

testValidGetAll

testInvalidGetAll

DataBase – implementacja (3)

```
class DataBase {

...

  // Execute query with one result
  public function getRow($q) {
    $pgq=@pg_query($this->conn,$q);
    if($pgq==NULL) throw new Exception("DataBaseInvalidQueryError");
    if($row=pg_fetch_assoc($pgq)) {
      return $row;
    } else return FALSE;
  }

...

};
```

testConnection

testValidQuery

testInvalidQuery

testValidGetRow

testInvalidGetRow

testValidGetAll

testInvalidGetAll

DataBase – implementacja (4)

```
class DataBase {

...
  // Execute query with many results
  public function getAll($q) {
    $res = array();
    $pgq=@pg_query($this->conn,$q);
    if($pgq==NULL) throw new Exception("DataBaseInvalidQueryError");
    while($row=pg_fetch_assoc($pgq)) $res[] = $row;
    return $res;
  }

};
```

- testConnection
- testValidQuery
- testInvalidQuery
- testValidGetRow
- testInvalidGetRow
- testValidGetAll
- testInvalidGetAll

CatModule - szkielet

```
require_once("CommandInt.php");

class Category {};

class CatList {};

class CatModDB {};

class CatModGUI {};

class CatModule implements CommandInt {
  public function action($req) {}
};
```

Category, CatList - szkielet

```
...
class Category {
  private $cid;
  private $cname;
  public function __construct($cn,$ci) {}
  public function getId() {}
  public function getName() {}
};

class CatList implements Iterator {
  private $list;
  public function __construct() {}
  public function add(Category $cat) {}
  public function current() {}
  public function next() {}
  public function key() {}
  public function valid() {}
  public function rewind() {}
  public function count() {}
};
...
```

UNIT TESTS - Category

testEmptyConstructor

testConstructorWithName

testCompleteConstructor

UNIT TESTS - CatList

testConstructor

testAddCategory

testIterator

Category - implementacja

```
class Category {
  private $cid;
  private $cname;

  public function __construct($cn=NULL,$ci=NULL) {
    $this->cid = $ci;
    $this->cname = $cn;
  }

  public function getId() {
    return $this->cid;
  }

  public function getName() {
    return $this->cname;
  }
```

UNIT TESTS - Category

testEmptyConstructor

testConstructorWithName

testCompleteConstructor

UNIT TESTS - CatList

testConstructor

testAddCategory

testIterator

CatList – implementacja (1)

```
class CatList implements Iterator {
  private $list;

  public function __construct() {
    $this->list = array();
  }

  public function add(Category $cat) {
    $this->list[] = $cat;
  }

  public function current() {}
  public function next() {}
  public function key() {}
  public function valid() {}
  public function rewind() {}

  public function count() {
    return count($this->list);
  }
};
```

UNIT TESTS - Category

testEmptyConstructor

testConstructorWithName

testCompleteConstructor

UNIT TESTS - CatList

testConstructor

testAddCategory

testIterator

CatList – implementacja (2)

```
class CatList implements Iterator {
  private $list;
  private $ptr;

  public function __construct() {
    $this->list = array();
    $this->ptr = 0;
  }

...

  public function current() { return $this->list[$this->ptr]; }

  public function next() { $this->ptr++; }

  public function key() { return $this->list[$this->ptr]->getId(); }

  public function valid() { return isset($this->list[$this->ptr]); }

  public function rewind() { $this->ptr = 0; }

....
```

UNIT TESTS - Category

testEmptyConstructor

testConstructorWithName

testCompleteConstructor

UNIT TESTS - CatList

testConstructor

testAddCategory

testIterator

CatModDB - szkielet

```
class CatModDB {

  public function getCat($id) {}

  public function addCat(Category $cat) {}

  public function getCatList() {}

  public function deleteCat($id) {}

  public function updateCat(Category $cat) {}

};
```

UNIT TESTS - CatModDB

- testDbGetCat
- testDbGetCatList
- testDbAddCat
- testDbDeleteCat
- testDbUpdateCat
- testDbAddCatDuplicateName
- testDbUpdateCatDuplicateName
- testDbGetCatInvalidId
- testDbDeleteCatInvalidId

CatModDB – implementacja (2)

```php
public function getCat($id) {
  $db = DataBase::instance();
  $res=$db->getRow("select * from bookshelf_catlist where id_c=$id;");
  return new Category($res["cname"],$res["id_c"]);
}

public function addCat(Category $cat) {
  $name = $cat->getName();
  $db = DataBase::instance();
  $query = "insert into bookshelf_catlist(cname) values ('$name');";
  $db->query($query);
}

public function getCatList() {
  $cl = new CatList();
  $db = DataBase::instance();
  $res=$db->getAll("select * from bookshelf_catlist;");
  foreach($res as $r) $cl->add(new Category($r["cname"],$r["id_c"]));
  return $cl;
}
```

UNIT TESTS - CatModDB

- testDbGetCat
- testDbGetCatList
- testDbAddCat
- testDbDeleteCat
- testDbUpdateCat
- testDbAddCatDuplicateName
- testDbUpdateCatDuplicateName
- testDbGetCatInvalidId
- testDbDeleteCatInvalidId

CatModDB – implementacja (3)

```
public function deleteCat($id) {
  $db = DataBase::instance();
  $query = "delete from bookshelf_catlist where id_c=$id;";
  $db->query($query);
}

public function updateCat(Category $cat) {
  $name = $cat->getName();
  $id = $cat->getId();
  $db = DataBase::instance();
  $query = "update bookshelf_catlist set cname='$name' where id_c=$id
  $db->query($query);
}
```

UNIT TESTS - CatModDB

testDbGetCat

testDbGetCatList

testDbAddCat

testDbDeleteCat

testDbUpdateCat

testDbAddCatDuplicateName

testDbUpdateCatDuplicateName

testDbGetCatInvalidId

testDbDeleteCatInvalidId

CatModDB – implementacja (4)

```php
public function getCat($id) {
   $db = DataBase::instance();
   $res=$db->getRow("select * from bookshelf_catlist where id_c=$id;");
   if($res==FALSE) throw new Exception("InvalidCategoryID");
   return new Category($res["cname"],$res["id_c"]);
  }

...

  public function deleteCat($id) {
   $db = DataBase::instance();
   $res = $db->getRow("select * from bookshelf_catlist where id_c=$id;");
   if($res==FALSE) throw new Exception("InvalidCategoryId");
   $db->query("delete from bookshelf_catlist where id_c=$id;");
  }
```

IT TESTS - CatModDB

- tDbGetCat
- tDbGetCatList
- tDbAddCat
- tDbDeleteCat
- tDbUpdateCat
- tDbAddCatDuplicateName
- tDbUpdateCatDuplicateName
- tDbGetCatInvalidId
- tDbDeleteCatInvalidId

CatModGUI - szablon

```
class CatModGUI {

  public function buttonAddCat() {}

  public function buttonUpdateCat() {}

  public function buttonRmoveCat() {}

  public function buttonListCat() {}

  public function formCategory() {}

  public function receiveCategory() {}

  public function showCatList() {}

};
```

Formularze typu „przycisk”

- Klasa CatModGUI zawiera szereg metod o nazwach „button*” - są to formularze złożone z jednego przycisku
- Takie same przyciski występowały już w klasie AccMod.
- Osobna implementacja każdego 1-przyciskowego formularza prowadzi do niepotrzebnego duplikowania kodu.
- Czas na **REFAKTORING**.

REFAKTORING

- Refaktoring – modyfikacja kodu bez zmiany jego funkcjonalności.
- W naszym przypadku – modyfikacja klasy AccMod i przeniesienie kodu generującego 1-przyciskowy formularz do klasy HtmlTools.
- W ten sposób uprościmy metodę AccMod::getModeButton i przeniesiemy kod wykorzystujący HTML do klasy HtmlTools.
- Dodatkowym „bonusem” będzie możliwość użycia tego samego kodu w kolejnych modułach.

AccMod::getModeButton

```
public function getModeButton() {
  $str = "<form action=".$_SERVER['PHP_SELF']." method=\"GET\">";
  if($_SESSION['AccMode']=="Edit") {
    $str.= "<input type=\"hidden\" name=\"cmd\" value=\"E-\">";
    $str.= "<input type=\"submit\" value=\"Edit mode on\"></form>";
  } else {
    $str.= "<input type=\"hidden\" name=\"cmd\" value=\"E+\">";
    $str.= "<input type=\"submit\" value=\"Edit mode off\"></form>";
  };
  return $str;
}
```

Refaktoryzacja i testy jednostkowe

- W celu zwiększenia bezpieczeństwa refaktoringu modyfikowany kod powinien być pokryty testami.
- Przygotujemy dodatkowe testy dla metody AccMod::getModeButton.
- Celem testów będzie weryfikacja podstawowych elementów generowanego kodu HTML.

AccModeTest

```
public function testEditOnButton() {
  AccMode::setMode(array("cmd"=>"E+"));
  $str = AccMode::getModeButton();
  $this->assertEquals(1,preg_match("/^<form.*form>\$/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"hidden\" name=\"cmd\" value=\"E-\">/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"submit\" value=\"Edit mode on\">/i",$str));
}

public function testEditOffButton() {
  AccMode::setMode(array("cmd"=>"E-"));
  $str = AccMode::getModeButton();
  $this->assertEquals(1,preg_match("/^<form.*form>\$/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"hidden\" name=\"cmd\" value=\"E\+\">/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"submit\" value=\"Edit mode off\">/i",$str));
}
```

HtmlTools

```
class HtmlTools {

...

  public function formButton($name,$fields) {

    return $str;
  }

};
```

Szablon metody formButton.

Metoda powinna zwracać formularz złożony z przycisku o nazwie $name i polach ukrytych zdefiniowanych w tablicy $fields (klucz – nazwa pola ukrytego, wartość – wartość pola ukrytego)

HtmlToolsTest

```
class HtmlToolsTest extends PHPUnit_Framework_TestCase {

 public function testFormButtonNoHidden() {
  $str = HtmlTools::formButton("test",array());
  $this->assertEquals(1,preg_match("/^<form.*form>\$/i",$str));
  $this->assertEquals(0,preg_match("/<input type=\"hidden\" name=/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"submit\" value=\"test\">/i",$str));
 }

 public function testFormButtonOneHidden() {
  $str = HtmlTools::formButton("test",array("h1"=>"v1"));
  $this->assertEquals(1,preg_match("/^<form.*form>\$/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"hidden\" name=\"h1\" value=\"v1\">/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"submit\" value=\"test\">/i",$str));
 }

 public function testFormButtonTwoHidden() {
  $str = HtmlTools::formButton("test",array("h1"=>"v1","h2"=>"v2"));
  $this->assertEquals(1,preg_match("/^<form.*form>\$/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"hidden\" name=\"h1\" value=\"v1\">/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"hidden\" name=\"h2\" value=\"v2\">/i",$str));
  $this->assertEquals(1,preg_match("/<input type=\"submit\" value=\"test\">/i",$str));
 }
```

formButton - implementacja

```
public function formButton($name,$fields) {
  $str = "<form action=".$_SERVER['PHP_SELF']." method=\"GET\">";
  foreach($fields as $k=>$v) {
      $str.= "<input type=\"hidden\" name=\"$k\" value=\"$v\">";
  };
  $str.= "<input type=\"submit\" value=\"$name\"></form>";
  return $str;
}
```

UNIT TESTS - HtmlTools

testFormButtonNoHidden

testFormButtonOneHidden

testFormButtonTwoHidden

Refaktoryzacja metody AccMode::getModeButton

```
public function getModeButton() {
   if($_SESSION['AccMode']=="Edit")
     $str = HtmlTools::formButton("Edit mode on",array("cmd"=>"E-"));
   else
     $str = HtmlTools::formButton("Edit mode off",array("cmd"=>"E+"));
   return $str;
 }
```

UNIT TESTS - AccMode

...

testEditOnButton

testEditOffButton

CatModGUI – szablon (usunięte metody button*)

```
class CatModGUI {

  public function formCategory() {}

  public function receiveCategory() {}

  public function showCatList() {}

};
```