MongoDB

PODSTAWOWE KONCEPCJE
API DLA JĘZYKA PHP
OPERACJE NA DOKUMENTACH

Waldemar Korłub

Wytwarzanie Aplikacji Internetowych
KASK ETI Politechnika Gdańska

Sterowniki PHP dla MongoDB

- Sterownik „mongo” (starszy)
  - Wersje MongoDB od 2.4 do 3.0 włącznie
    - Brak obsługi nowszych: 3.2, 3.4, 3.6…
  - Wersje PHP do 5.3 do 5.6 włącznie
    - Niedostępny dla PHP 7.x
- Sterownik „mongodb” (nowszy)
  - Wszystkie wersje MongoDB od 2.4 wzwyż
  - Wszystkie wersje PHP od 5.4 wzwyż
- Tabela kompatybilności wersji MongoDB z wersjami sterowników i wersjami PHP:
  https://docs.mongodb.org/ecosystem/drivers/php/

Sterowniki PHP dla MongoDB

- Sterowniki różnią się pod względem API
  - Należy uważać szukając przykładów/tutoriali/rozwiązań problemów w Internecie
  - Przykłady ze strony przedmiotu (eNauczanie) dotyczą nowszego sterownika „mongodb” i PHP 7.x
    - Maszyna wirtualna zawiera sterownik „mongodb” i PHP 7
- Sterownik „mongodb” oferuje niskopoziomowy interfejs do komunikacji z bazą danych
- Biblioteka mongo-php-library oferuje wygodniejsze dla dewelopera wysokopoziomowe API
  - Należy dodać ją do zależności projektu

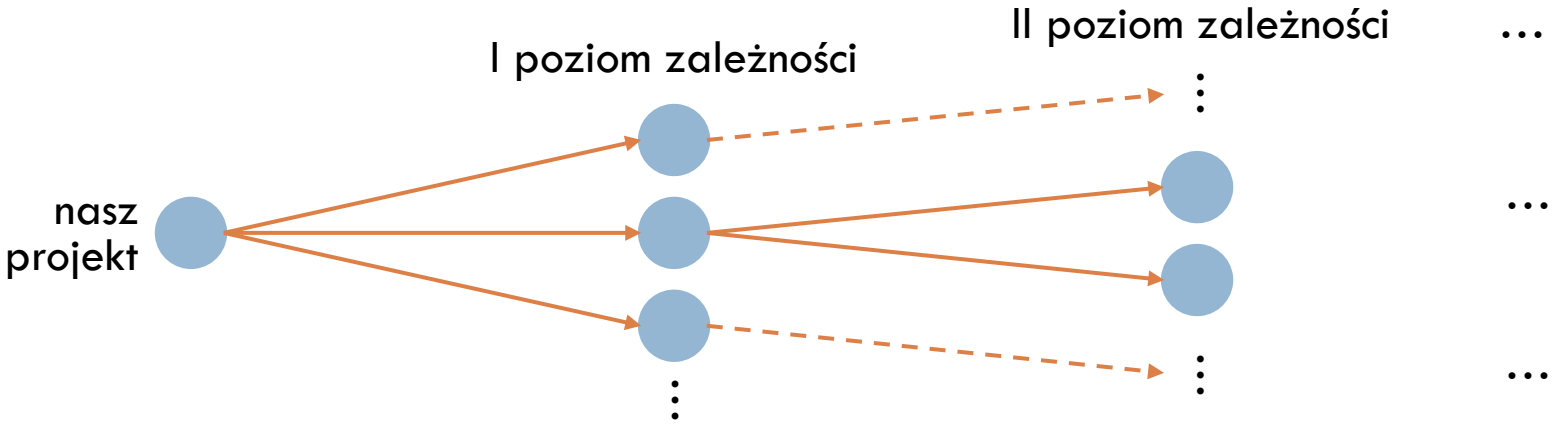
Composer

Zarządzanie zależnościami projektu

Zależności projektu

- Zależności naszego projektu zazwyczaj posiadają własne zależności
  - My używamy bibliotek, aby przyspieszyć swoją pracę
  - Autorzy bibliotek używają innych bibliotek, aby przyspieszyć swoją

- Powstaje drzewo zależności



Zależności projektu

- Zależnościami można zarządzać ręcznie
  1. Google.com – szukamy strony projektu
  2. Wchodzimy na stronę biblioteki
  3. Dział Downloads
  4. Pobieramy wersję dystrybucyjną (np. archiwum zip)
  5. Rozpakowujemy
  6. Kopiujemy pliki biblioteki do naszego projektu
  7. I tak dla każdej zależności w drzewie…
- Ręczne zarządzanie zależnościami jest czasochłonne, kłopotliwe i po prostu nudne

Automatyczne zarządzanie zależnościami

- Narzędzie Composer automatyzuje zarządzanie zależnościami projektu
- Należy zdefiniować zależności I poziomu, przykładowo:

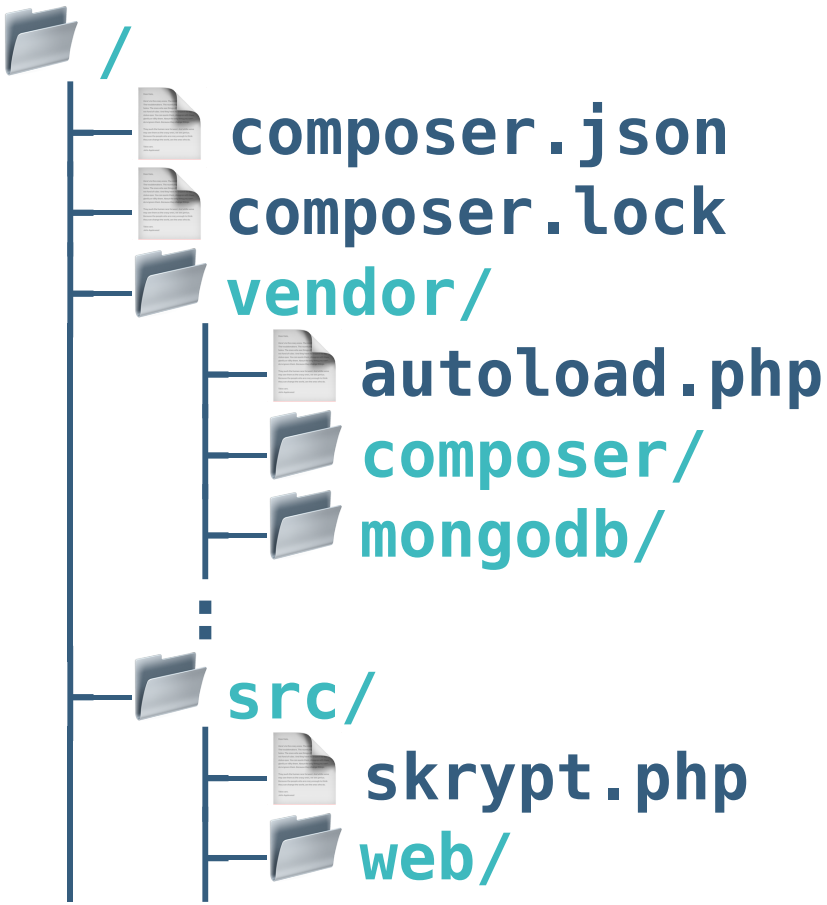
```
$ composer require mongodb/mongodb "^1.1"
```

- Nazwa wskazanej biblioteki i jej wersja zostają dopisane do pliku composer.json
  - composer.json – zawiera listę zależności I poziomu

Automatyczne zarządzanie zależnościami

- Composer buduje drzewo zależności projektu
- …a następnie automatycznie pobiera wszystkie wymagane biblioteki
  - Podkatalog vendor – zawiera zewnętrzne zależności projektu (*3rd party*)
  - Skrypt vendor/autoload.php – umożliwia automatyczne ładowanie klas z zewnętrznych bibliotek
- Zapisany zostaje tzw. *lockfile* (composer.lock)
  - *lockfile* zawiera informacje o dokładnych wersjach wszystkich pobranych bibliotek
    - Również zależności poziomu II oraz dalszych

Struktura projektu



główny katalog projektu

biblioteki *3rd party*

nasz kod źródłowy

DocumentRoot

MongoDB

MongoDB

- Baza typu NoSQL
  - Rezygnacja z gwarancji zapewnianych przez bazy relacyjne na rzecz *innych* właściwości
- Dane o dynamicznej strukturze
  - Nie występuje z góry zdefiniowany schemat dla przechowywanych informacji
- Baza dokumentowa
  - *Dokument* jako podstawowy nośnik informacji

Dokument

- Dokument: zbiór par klucz → wartość
- Możliwe wartości:
  - Dane skalarne
  - Tablice
  - Inne dokumenty – dokumenty zagnieżdżone

Dokument – reprezentacja JSON


```
{
  "nazwa": "Laptop XYZ",
  "producent": "ABC123",

  "porty": ["4 USB", "HDMI", "Ethernet"],

  "specyfikacja": {
    "CPU": "Intel i7",
    "RAM": "8 GB",
    "HDD": "1 TB"
  }
}
```

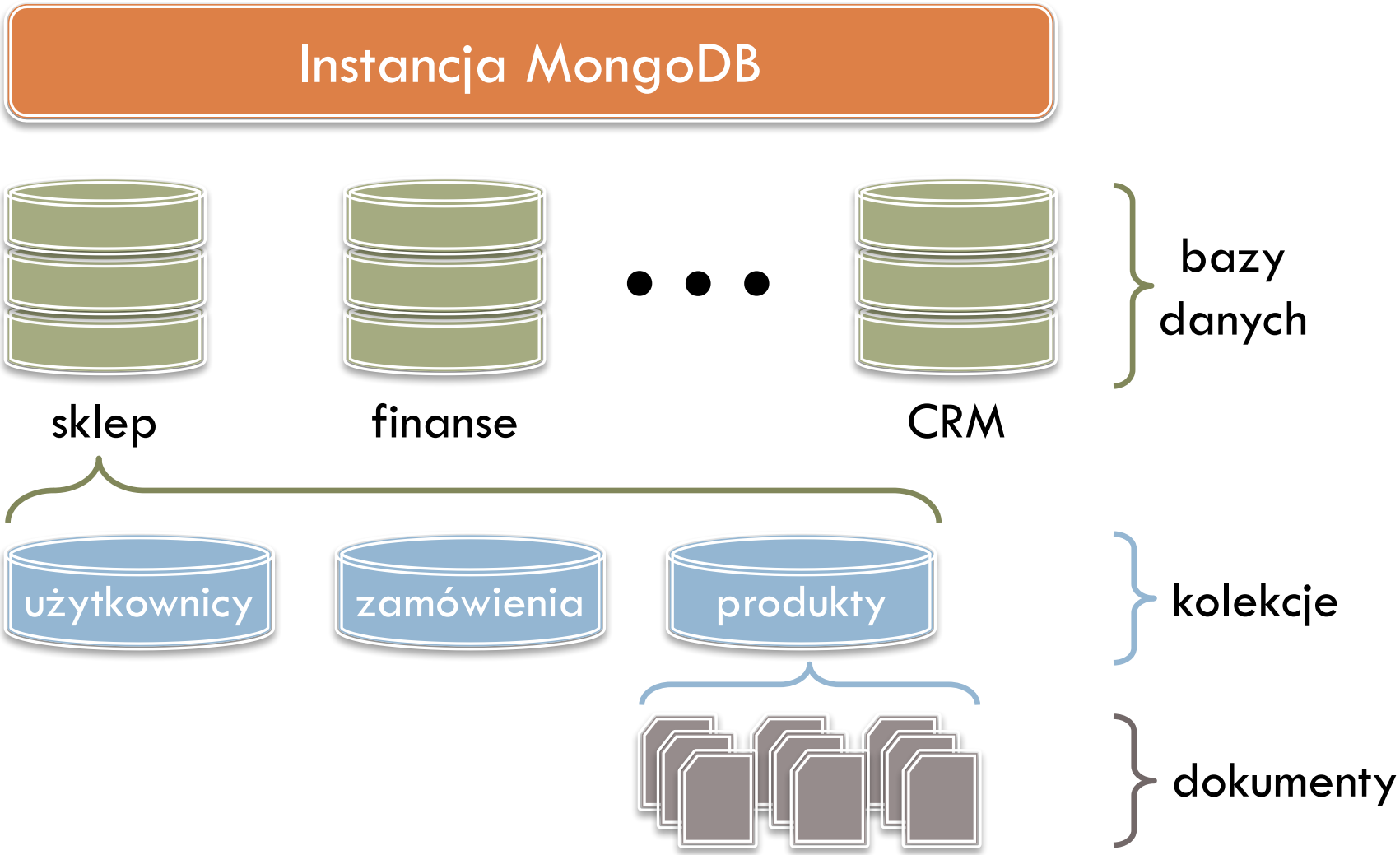
Dokument – reprezentacja w PHP


```php
<?php
$dokument = [
  "nazwa" => "Laptop XYZ",
  "producent" => "ABC123",

  "porty" => ["4 USB", "HDMI", "Ethernet"],

  "specyfikacja" => [
    "CPU" => "Intel i7",
    "RAM" => "8 GB",
    "HDD" => "1 TB"
  ]
];
```

Kolekcja, baza danych, instancja

- Zbiór dokumentów tego samego typu tworzy *kolekcję*, np.:
  - Kolekcja produktów w sklepie
  - Kolekcja zamówień
- Wiele kolekcji składa się na *bazę danych*:
  - Logicznie powiązane ze sobą kolekcje, np. wszystkie kolekcje z danymi sklepu internetowego
- Pojedyncza *instancja* MongoDB może hostować wiele baz danych
  - Osobne bazy danych pozwalają odseparować od siebie dane różnych aplikacji

Dokument, kolekcja, baza danych, instancja



Dynamiczna struktura danych

- Nie trzeba definiować kolekcji – zostaną automatycznie dodane przy pierwszym użyciu
- Nie trzeba definiować *a priori* struktury dokumentów
  - Dokumenty budowane *ad hoc* w zależności od kontekstu
  - Nie trzeba dostosowywać danych do struktury bazy – to baza dostosowuje się do otrzymywanych danych
- W jednej kolekcji mogą znajdować się dokumenty o różnej strukturze
  - np. z polami opcjonalnymi

API w PHP do pracy z MongoDB

Połączenie z MongoDB


```
function get_db()
{
    $mongo = new MongoDB\Client(
        "mongodb://localhost:27017/wai",
        [
            'username' => 'wai_web',
            'password' => 'w@i_w3b',
        ]);

    $db = $mongo->wai;

    return $db;
}
```

Zapisywanie dokumentów


```php
$db = get_db();

//dokument opisujący produkt w sklepie:
$product = [
    'name' => 'Laptop XYZ',
    'price' => 2999,
    'manufacturer' => 'ABC',
    'description' => 'Ładny laptop.'
];

//zapis do kolekcji products:
$db->products->insertOne($product);
```

Kolekcja zostanie dodana, jeśli wcześniej nie istniała

Identyfikator dokumentu

- Każdy dokument w bazie MongoDB posiada identyfikator w postaci pola **_id**
- Jeśli pole **_id** nie zostanie określone jawnie, baza doda je automatycznie i przydzieli unikalny identyfikator:

```
{
  "_id": ObjectId("56059e41b80de4f3478b4567"),
  "name": "Laptop XYZ",
  "price": NumberLong(2999),
  "manufacturer": "ABC",
  "description": "Ładny laptop."
}
```

Wyszukiwanie dokumentów


```php
$db = get_db();

//wyszukiwanie bez kryteriów dopasowania,
//zwraca wszystkie elementy:
$products = $db->products->find();

foreach ($products as $product) {
    echo $product['name'] . '<br/>';
}
```

Zwrócony kursor umożliwia iterowanie po wynikach

Wyszukiwanie dokumentów

- Wyniki można zawęzić wyszukując *poprzez przykład* (ang. *query by example*)
- Należy przygotować *przykład* dokumentu, który ma zostać odnaleziony
  - Dokument, który przypomina spodziewane rezultaty wyszukiwania
- Tak przygotowany przykład należy przekazać jako kryterium wyszukiwania

Wyszukiwanie *by example*

- Cel: wyszukanie towarów w cenie 2999 PLN od producenta ABC123
- Przykład dokumentu:

```
$query = [
    'price' => 2999,
    'manufacturer' => 'ABC123',
];
```

Koniunkcja warunków

Pozostałe pola dowolnej wartości

- Wyszukanie pasujących dokumentów:

```
$products = $db->products->find($query);
```

Wyszukiwanie na podstawie _id


```php
use MongoDB\BSON\ObjectID;
//...
$id = "56059e41b80de4f3478b4567";

$query = [
    '_id' => new ObjectId($id)
];

$product = $db->products->findOne($query);
```

Identyfikator w postaci tekstowej

Identyfikator bazodanowy

Oczekiwany pojedynczy rezultat

Operatory wyszukiwania

- Poza dopasowaniem dokładnych wartości pól dokumentów, przydają się też dodatkowe operatory
  - np. wyszukanie towarów w cenie 2500 – 3000 PLN

```
$query = [
    'price' => [
        '$gt' => 2500,
        '$lt' => 3000
    ]
];

$products = $db->products->find($query);
```

Operator alternatywy


```
$products = $db->products->find([
    '$or' => [
        ['price' => ['$lt' => 3000]],
        ['manufacturer' => 'Apple'],
    ]
]);
```

- Wyszukanie produktów:

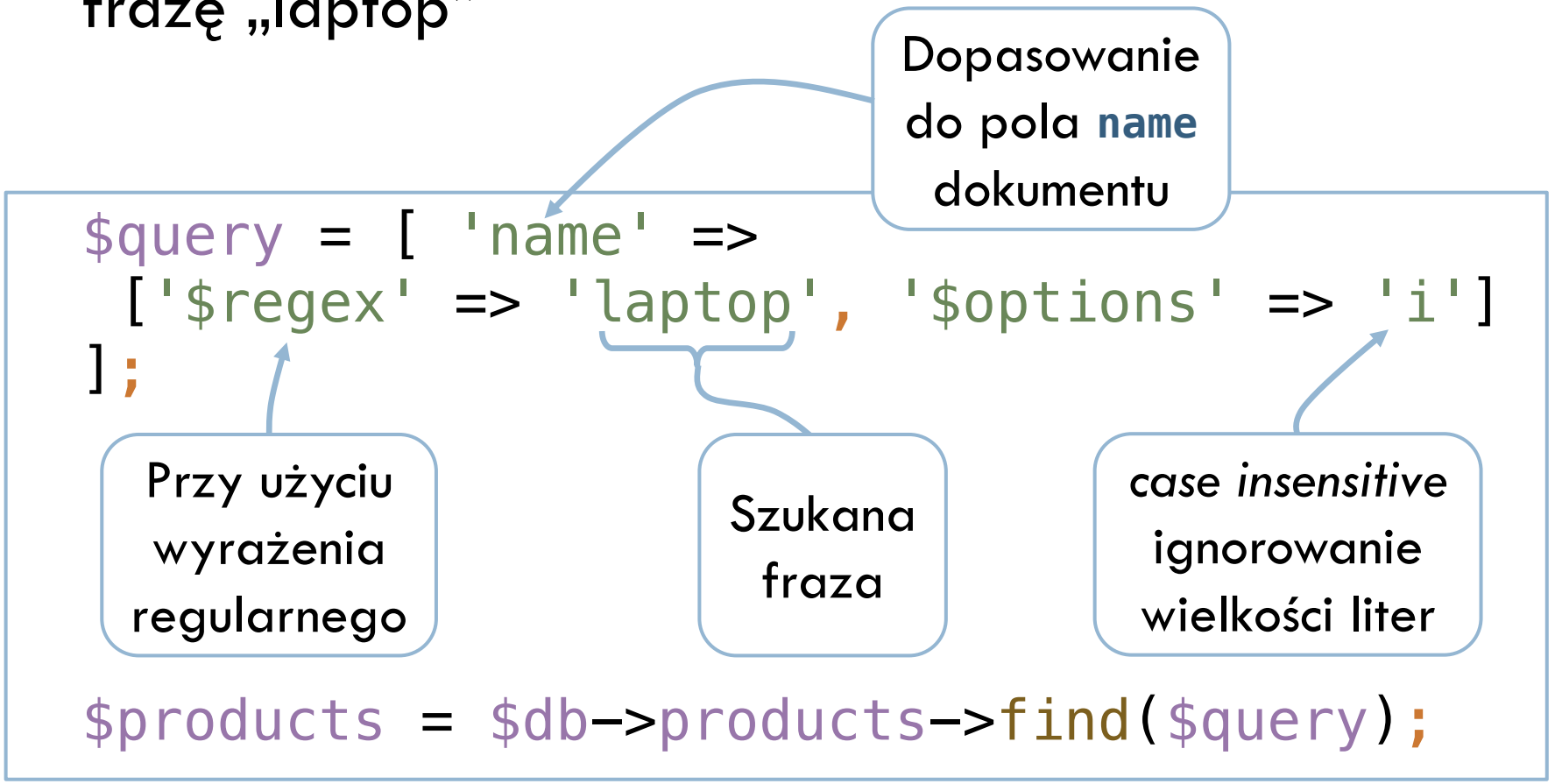
  których cena nie przekracza 3000 zł

  LUB

  wyprodukowanych przez firmę Apple

Wyszukiwanie wyrażeniem regularnym

- Cel: wyszukać produkty, których nazwa zawiera frazę „laptop”



Warunki z dokumentami zagnieżdżonymi

- Warunki mogą odnosić się do wartości pól dokumentów zagnieżdżonych
- Składnia *dot notation* pozwala trawersować w głąb dokumentów zagnieżdżonych

```
$product = [
 "nazwa" => "Laptop",
 //...
 "specs" => [
    "CPU" => "Intel i7",
    "RAM" => "8 GB",
    "HDD" => "1 TB"
 ]
];
```

```
$results = $db->products->
              find(['specs.CPU'=>'Intel i7']);
```

dot notation

Ograniczenie liczby wyników


```
$opts = [
    'skip' => 10,
    'limit' => 5
];
```

Pominięcie pierwszych 10 wyników

Ograniczenie do maksymalnie 5 wyników

```
$products=$db->products->find($query, $opts);

foreach ($products as $product) {
    //...
}
```

Stronicowanie


```php
$page = 3;
$pageSize = 3;
$opts = [
    'skip' => ($page - 1) * $pageSize,
    'limit' => $pageSize
];
$users = $db->users->find([], $opts);

foreach ($users as $user) {
    //...
}
```

#	Imię	Nazwisko
7	Imiesław	Nazwiskowy
8	Waldemar	Korłub
9	Nazwisław	Imieński

« 1 2 3 4 5 »

Sortowanie

- Produkty w kolejności od najtańszego do najdroższego:

```
$opts = ['sort' => ['price' => 1]];
$products = $db->products->find([], $opts);
```

Według ceny rosnąco

- Sortowanie wielokryterialne:

```
$opts = [
    'sort' => [
        'price' => -1,
        'name' => 1
    ]
];
$products = $db->products->find([], $opts);
```

Według ceny malejąco

Jeśli ceny są takie same: według nazwy rosnąco

Aktualizacja: podmiana całego dokumentu


```
$id = '5605b1e9b80de4f1478b4571';
$query = ['_id' => new ObjectId($id)];

//pobranie dokumentu:
$product = $db->products->findOne($query);
//nowa cena:
$product['price'] = 2699;

//podmiana dotychczasowego dokumentu na nowy:
$db->products->replaceOne($query, $product);
```

Aktualizacja: wiele dokumentów


```
$query = [
    'name' => [
      '$regex' => 'laptop',
      '$options' => 'i’
    ]
];

$newVals = ['price' => 999];

$db->products->
     updateMany($query,['$set' => $newVals]);
```

Aktualizowane pola

$set: aktualizuje wskazane pola, inne pozostawia niezmienione

Usuwanie dokumentów


```php
use MongoDB\BSON\ObjectID;

$id = '5605b1e9b80de4f1478b4571';
$query = ['_id' => new ObjectId($id)];

$db->products->deleteOne($query);
```

Przykład: 02-mongodb

Pytania?