Ocena skuteczności wykorzystania wybranych nowoczesnych technologii na podstawie doświadczeń wynikających z utworzenia internetowej platformy ogłoszeniowej

Artur Nowakowski

Platforma ogłoszeniowa ze sprzętem rowerowym

Podstawowa funkcjonalność:

- Dodawanie ogłoszeń
- Edycja ogłoszeń
- Wyszukiwarka ogłoszeń
- Formularz kontaktowy ze sprzedającym
- Panel administracyjny
- Moderacja ogłoszeń

Technologie, frameworki

W swoim projekcie użyję następujących technologii:

- Grails
- AngularJS
- HTML, CSS, Bootstrap
- Representational State Transfer (REST)
- JSON
- Maven
- GIT

Grails

Grails to opensource'owy framework do tworzenia aplikacji internetowych działających pod kontrolą Javy. Dodatkową zaletą jest oparcie frameworka na skryptowym języku Groovy, który znacznie upraszcza np. działanie na kolekcjach.

, Z paczki' dostajemy m.in:

- wsparcie dla wzorca MVC,
- GORM – Grails Object Relational Mapping – technika konwersji danych między dwoma niekompatybilnymi systemami. W efekcie działania GORM, obiekty w kodzie są mapowane na obiekty w bazie danych.
- Wspracie dla testów
- Serwer aplikacyjny.
- Spora baza wtyczek.

Grails - przykład działania

Klasa domenowa:

```
class Advert {
    AdvertKey key=AdvertKey.generate()
    String title
    String description
    SellerKey sellerKey;
    CategoryKey categoryKey

    static  mapping = {
        key column: 'bKey'
    }
    static constraints = {
        key unique: true, blank: false, empty:false
        title nullable:false
        description nullable:false
        sellerKey nullable: true
    }
}
```

Kontroler:

```
class AdvertController {

    def create(){
        Advert advert=new Advert(
            title: "Sprzedam rower",
            description: "Jest super!"
        ).save()
    }

}
```

AngularJS

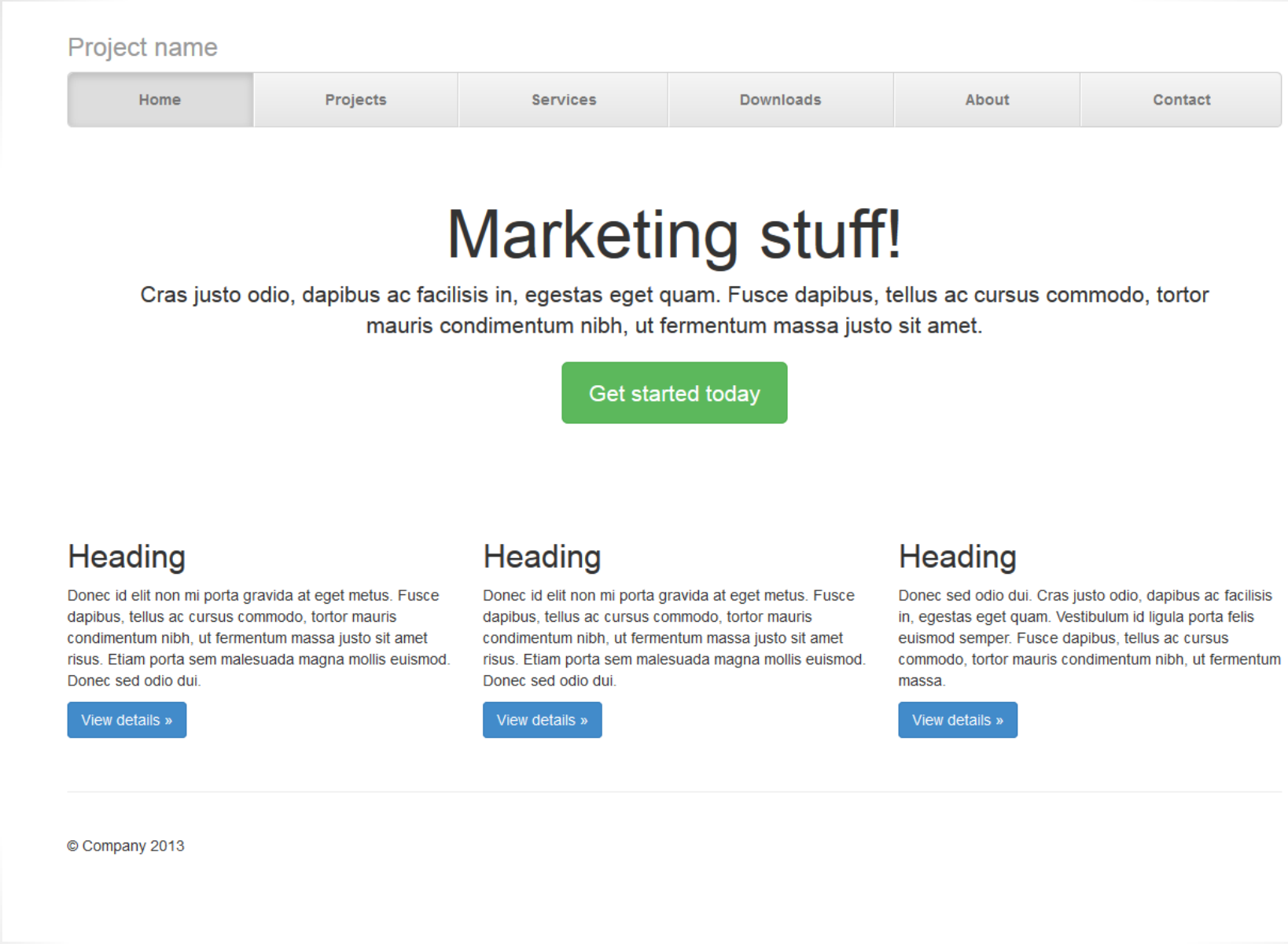
AngularJS - otwarta biblioteka języka JavaScript, stworzona przez Google, wspomagająca tworzenie i rozwój aplikacji internetowych na pojedynczej stronie. Zadaniem biblioteki jest wdrożenie wzorca Model-View-Controller (MVC) do aplikacji internetowych, aby ułatwić ich rozwój i testowanie.

W Angularze mamy m.in możliwość rozszerzania funkcjonalności standardowych komponentów HTML. Angular doskonale sprawdza się do wyświetlania danych dostarczonych mu w formacie JSON. Znacząco przyspiesza np. wyświetlanie listy obiektów w formie tabeli, a także ułatwia walidację danych w formularzach.

Bootstrap

Bootstrap – lekki prosty i przyjamny framework do tworzenia frontendu.

- Kolumnowy szablon – ułatwia tworzenie witryny dla różnych rozdzielczości
- Nowoczesny desgin
- Ładne komponenty (przyciski, listy, pola formualrzy itp.)
- Przydatne operacje w JavaScript. (Okna modalne, rozwijalne listy itp.)


Project name
Home
Projects
Services
Downloads
About
Contact
Marketing stuff!
Cras justo odio, dapibus ac facilisis in, egestas eget quam. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet.
Get started today
Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.
View details »
Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.
View details »
Heading
Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa.
View details »
© Company 2013

REST

Wzorzec architektury oprogramowania, wprowadzającym dobre praktyki tworzenia aplikacji webowych. Jest szczególnie przydatny do tworzenia spójnego, czytelnego API programu.

Klasyczne zapytanie do serwera:
http://example.com/article?id=1234&format=print
Wywołanie REST:
http://example.com/article/1234/print

API RESTOWE powinno określać działanie metod http:
GET – pobranie listy elementów, bądź konkretnego obiektu
POST – Utworzenie elementu
PUT – Aktualizacja elementu
DELETE – Usunięcie elementu.

JSON

Lekki, tekstowy format wymiany danych komputerowych. Znakomicie nadaje się do komunikacji warstwy serwerowej z widokiem. Jest niezależny od języka programowania. Dane dostarczone w formacie JSON, są bardzo proste do przetworzenia.

Pretty | Raw | Preview | JSON | XML


```
[
    {
        "key": "deal-508d3732-93d3-41a3-998c-e6ebc408d8eb",
        "title": "Advert Title 1",
        "description": "Advert description 1",
        "category": {
            "name": "Downhill bikes",
            "key": "ctgr-ade2ec36-f37a-40c1-8c7f-3313b8196aae"
        }
    },
    {
        "key": "deal-ee199a37-5616-4706-a9f9-527d2af6622d",
        "title": "Advert Title 2",
        "description": "Advert description 2",
        "category": {
            "name": "Rear",
            "key": "ctgr-ca8e5b5b-94fb-4ec3-b8c0-6d8d0b9fab81"
        }
    },
    {
        "key": "deal-c3e02c5d-b460-413e-939e-1c7ac62d2103",
        "title": "Advert Title 3",
        "description": "Advert description 3",
        "category": {
            "name": "Rear",
            "key": "ctgr-ca8e5b5b-94fb-4ec3-b8c0-6d8d0b9fab81"
        }
    }
]
```

Maven

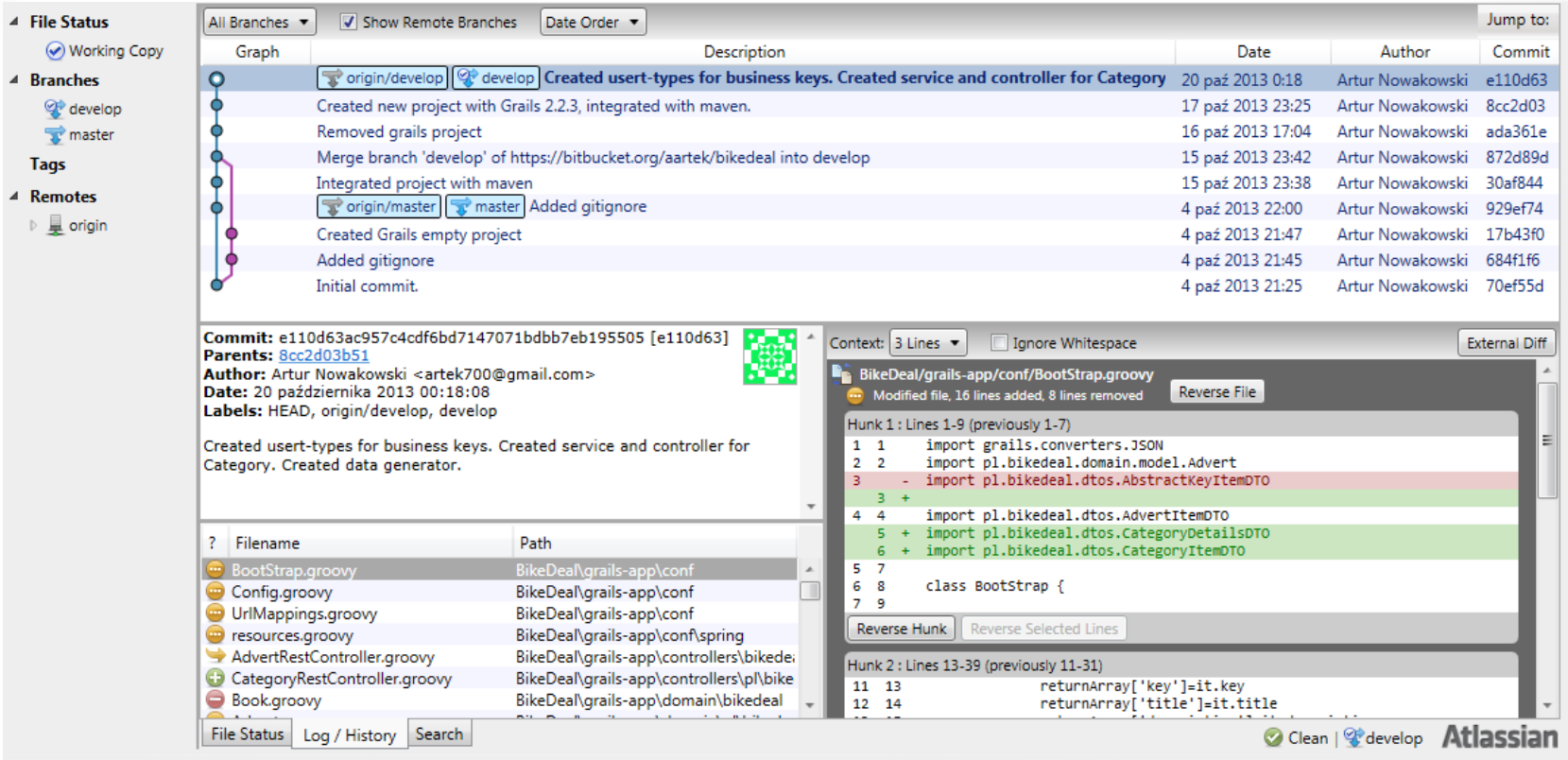
narzędziem automatyzującym budowę oprogramowania na platformę Java.

- Proste zarządzanie zależnościami w projekcie (zależności między modułami, zarządzanie bibliotekami)
- Ułatwione budowanie paczek dystrybucyjnych, np. pliku WAR, który może zostać później umiejscowiony na serwerze.
- Uruchamianie aplikacji
- Uruchamianie testów.

GIT

System kontroli wersji pozwalający nam m.in. na:

- Trzymanie aplikacji na zewnętrznym repozytorium
- Pełną historie zmian w kodzie aplikacji
- Rozwiązywanie konfliktów w kodzie jednocześnie zmienionym przez kilku programistów
- Tworzenie odgałęzień od głównego projektu
- Wydawanie wersji

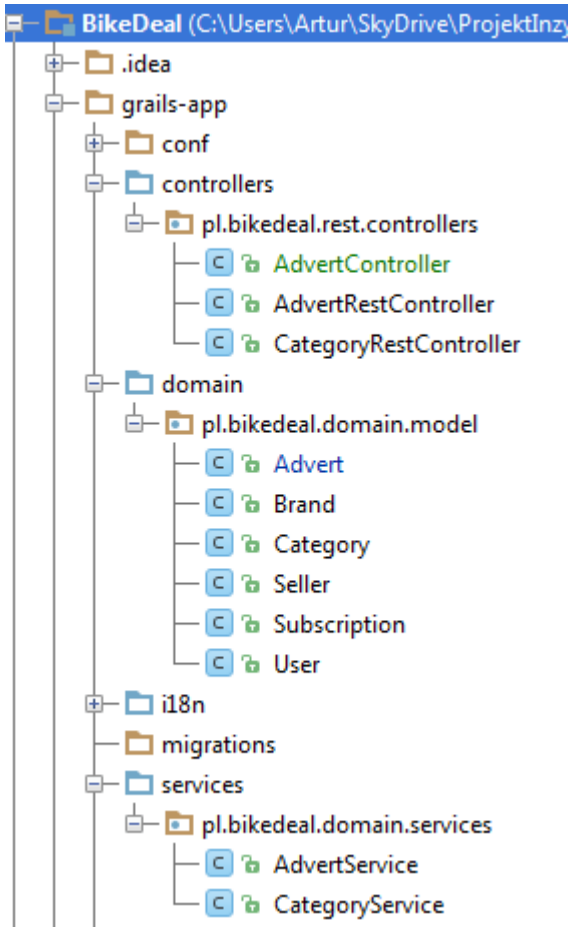

File Status
Working Copy
Branches
develop
master
Tags
Remotes
origin
All Branches
Show Remote Branches
Date Order
Jump to:
Graph
Description
Date
Author
Commit
origin/develop
develop
Created usert-types for business keys. Created service and controller for Category
20 paź 2013 0:18
Artur Nowakowski
e110d63
Created new project with Grails 2.2.3, integrated with maven.
17 paź 2013 23:25
Artur Nowakowski
8cc2d03
Removed grails project
16 paź 2013 17:04
Artur Nowakowski
ada361e
Merge branch 'develop' of https://bitbucket.org/aartek/bikedeal into develop
15 paź 2013 23:42
Artur Nowakowski
872d89d
Integrated project with maven
15 paź 2013 23:38
Artur Nowakowski
30af844
origin/master
master
Added gitignore
4 paź 2013 22:00
Artur Nowakowski
929ef74
Created Grails empty project
4 paź 2013 21:47
Artur Nowakowski
17b43f0
Added gitignore
4 paź 2013 21:45
Artur Nowakowski
684f1f6
Initial commit.
4 paź 2013 21:25
Artur Nowakowski
70ef55d
Commit: e110d63ac957c4cdf6bd7147071bdbb7eb195505 [e110d63]
Parents: 8cc2d03b51
Author: Artur Nowakowski <artek700@gmail.com>
Date: 20 października 2013 00:18:08
Labels: HEAD, origin/develop, develop
Created usert-types for business keys. Created service and controller for Category. Created data generator.
Context: 3 Lines
Ignore Whitespace
External Diff
BikeDeal/grails-app/conf/BootStrap.groovy
Modified file, 16 lines added, 8 lines removed
Reverse File
Hunk 1 : Lines 1-9 (previously 1-7)
1 1 import grails.converters.JSON
2 2 import pl.bikedeal.domain.model.Advert
3 - import pl.bikedeal.dtos.AbstractKeyItemDTO
3 +
4 4 import pl.bikedeal.dtos.AdvertItemDTO
5 + import pl.bikedeal.dtos.CategoryDetailsDTO
6 + import pl.bikedeal.dtos.CategoryItemDTO
5 7
6 8 class BootStrap {
7 9
Reverse Hunk
Reverse Selected Lines
Hunk 2 : Lines 13-39 (previously 11-31)
11 13 returnArray['key']=it.key
12 14 returnArray['title']=it.title
? Filename Path
BootStrap.groovy BikeDeal\grails-app\conf
Config.groovy BikeDeal\grails-app\conf
UrlMappings.groovy BikeDeal\grails-app\conf
resources.groovy BikeDeal\grails-app\conf\spring
AdvertRestController.groovy BikeDeal\grails-app\controllers\bikede
CategoryRestController.groovy BikeDeal\grails-app\controllers\pl\bike
Book.groovy BikeDeal\grails-app\domain\bikedeal
File Status
Log / History
Search
Clean | develop
Atlassian

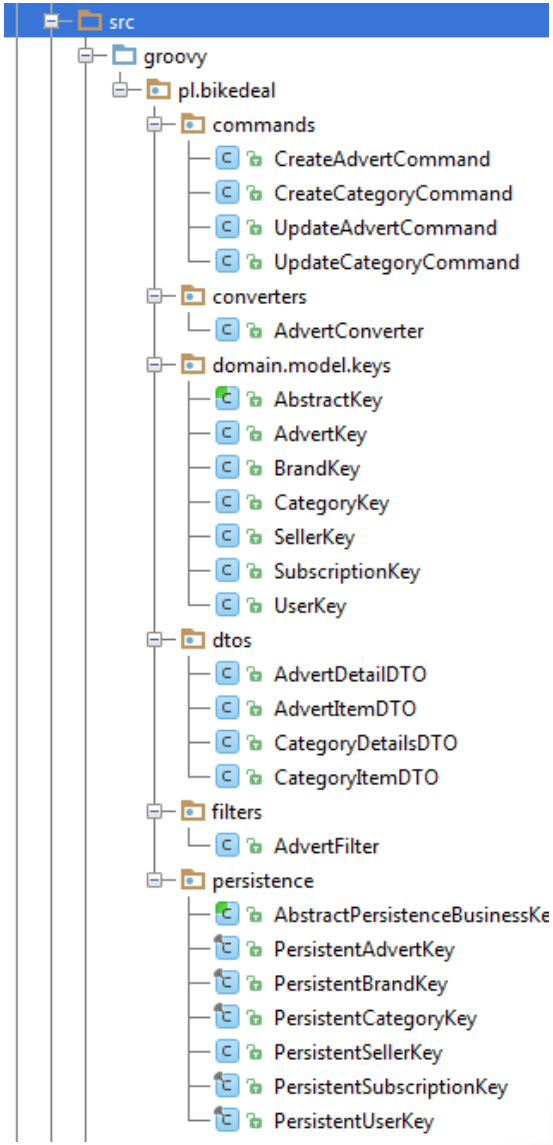
Projekt aplikacji

Moja aplikacja będzie podzielona na dwie główne warstwy:

- Frontend:
  - Strona internetowa stworzona przy użyciu HTMLa, Bootstrap'a i AngularJS.
  - Intefejs użytkownika, formularze, listy, czyli wszystko to co użytkownik ma zobaczyć.
- Backend
  - Część serwerowa, tu odbywają się wszystkie operacje przetwarzania danych, takie jak zapis obiektów z bazy danych, odczyt, czy edycja

Struktura aplikacji

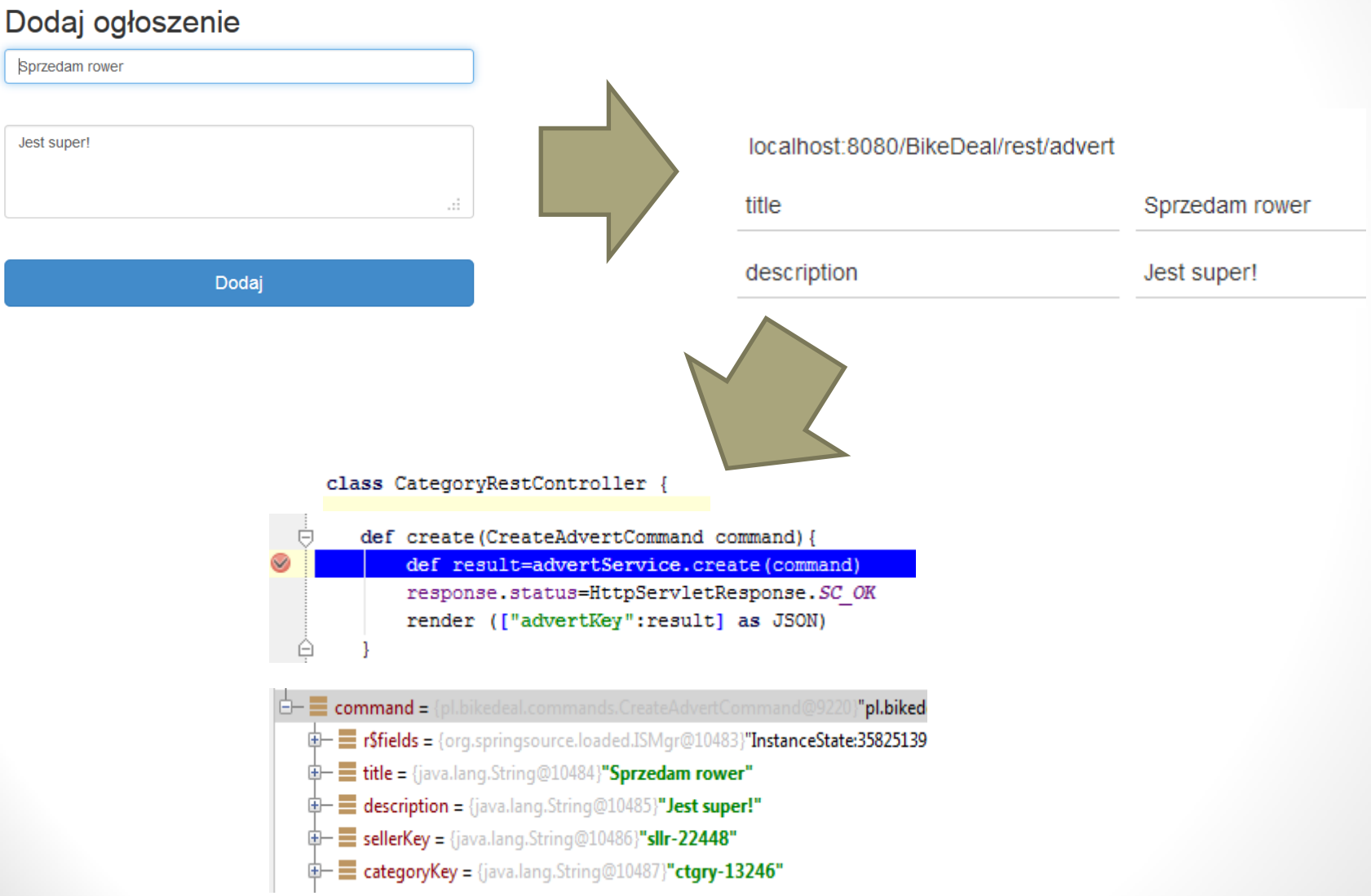




```groovy
class UrlMappings {

    static mappings = {
        "/$controller/$action?/$id?"{
            constraints {
                // apply constraints here
            }
        }

        "/"(view:"/index")
        "500"(view:'/error')

        //ADVERT
        "/rest/advert"(controller: "advertRest", parseRequest: true){
            action=[GET:"list", POST:"create", PUT:"update"]
        }

        //CATEGORY
        "/rest/category"(controller:"categoryRest", parseRequest: true){
            action=[GET:"list",POST:"create",PUT:"update"]
        }

        "/rest/category/$key"(controller:"categoryRest", parseRequest: true){
            action=[GET:"get"]
        }

        "/rest/category/subcategories/$key"(controller:"categoryRest", parseReq
            action=[GET:"subcategories"]
        }

    }
}
```

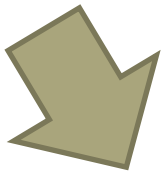
```groovy
JSON.registerObjectMarshaller(AdvertItemDTO){
    def returnArray=[:]
    returnArray['key']=it.key
    returnArray['title']=it.title
    returnArray['description']=it.description
    returnArray['category']=it.categoryItem
    return returnArray
}

JSON.registerObjectMarshaller(CategoryItemDTO){
    def returnArray=[:]
    returnArray['name']=it.name;
    returnArray['key']=it.key;
    return returnArray
}

JSON.registerObjectMarshaller(CategoryDetailsDTO){
    def returnArray=[:]
    returnArray['key']=it.key;
    returnArray['name']=it.name;
    returnArray['description']=it.description
    return returnArray
}
```

Komunikacja UI z serwerem



Komunikacja UI z serwerem

```
class AdvertService {

    def list(AdvertFilter filter){...}

    def create(CreateAdvertCommand command){
        Advert advert=new Advert();
        command.bindTo(advert);
        advert.save(flush:true, failOnError: true)
        return advert.key
    }
```



37	0	ctgr-ade2ec36-f37a-40c1-8c7f-3313b8196aae	Advert description 37	deal-d7d2348e-6d0c-485d-89a7-bbb90e653e42	*null*	Advert Title 37
38	0	ctgr-5c0705a2-9913-4e7f-ba99-a9aecbff57ec	Advert description 38	deal-73a1ceba-b8b7-44d3-a6bd-775ece9ede81	*null*	Advert Title 38
39	0	ctgr-ade2ec36-f37a-40c1-8c7f-3313b8196aae	Advert description 39	deal-ba1d2a8d-f611-4432-b866-a4846bf549dd	*null*	Advert Title 39
40	0	ctgr-5c0705a2-9913-4e7f-ba99-a9aecbff57ec	Advert description 40	deal-47bd872b-2158-4cec-afa7-d35ca17823f2	*null*	Advert Title 40
43	0	ctgry-13246	Jest super!	deal-8da7db9a-ed0c-478e-beef-197fc0e61fdf	sllr-22448	Sprzedam rower

(41 rows, 3 ms)

Body | Headers (4) | STATUS 200 OK | TIME 318613 ms

Pretty | Raw | Preview | JSON | XML

```
{
    "advertKey": {
        "key": "deal-8da7db9a-ed0c-478e-beef-197fc0e61fdf"
    }
}
```

KONIEC