SKRIPSI

PORTING PHP MENJADI JAVA/PLAY FRAMEWORK (STUDI KASUS KIRI *DASHBOARD SERVER SIDE*)

TOMMY ADHITYA THE

NPM: 2012730031

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2015

UNDERGRADUATE THESIS

PORTING PHP TO JAVA/PLAY FRAMEWORK (CASE STUDY KIRI DASHBOARD SERVER SIDE)

TOMMY ADHITYA THE

NPM: 2012730031

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2015

LEMBAR PENGESAHAN

PORTING PHP MENJADI JAVA/PLAY FRAMEWORK (STUDI KASUS KIRI *DASHBOARD SERVER SIDE*)

TOMMY ADHITYA THE

NPM: 2012730031

Bandung, 17 September 2015

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
Pascal Alfadian, M.Com.	**«pembimbing pendamping/2»**
Ketua Tim Penguji	**Anggota Tim Penguji**
«penguji 1»	**«penguji 2»**

Mengetahui,

Ketua Program Studi

Mariskha Tri Aditia, PDEng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PORTING PHP MENJADI JAVA/PLAY FRAMEWORK (STUDI KASUS KIRI *DASHBOARD SERVER SIDE*)

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 17 September 2015

Meterai

Tommy Adhitya The
NPM: 2012730031

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini. . . ?»

KATA PENGANTAR

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Bandung, September 2015

Penulis

DAFTAR ISI

DAFTAR GAMBAR

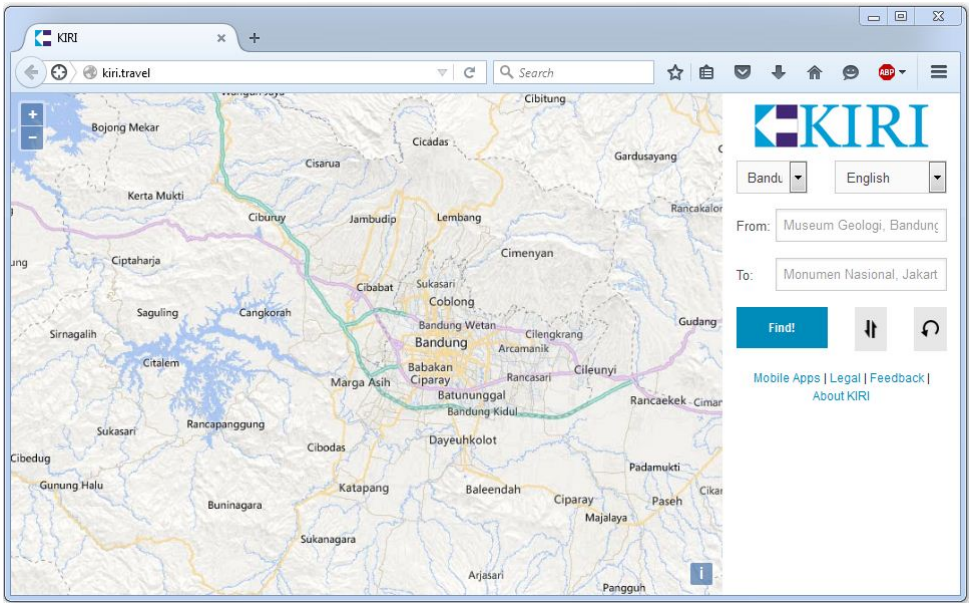
DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

KIRI[1] (gambar 1.1) merupakan situs web untuk membantu pengguna menemukan rute transportasi umum ke tempat tujuannya. Dengan memasukkan lokasi awal serta lokasi tujuan pengguna tersebut, situs web KIRI akan memberikan langkah-langkah (contoh: berjalan sejauh berapa meter, menggunakan angkot, dan sebagainya) tercepat untuk sampai ke lokasi tujuan.

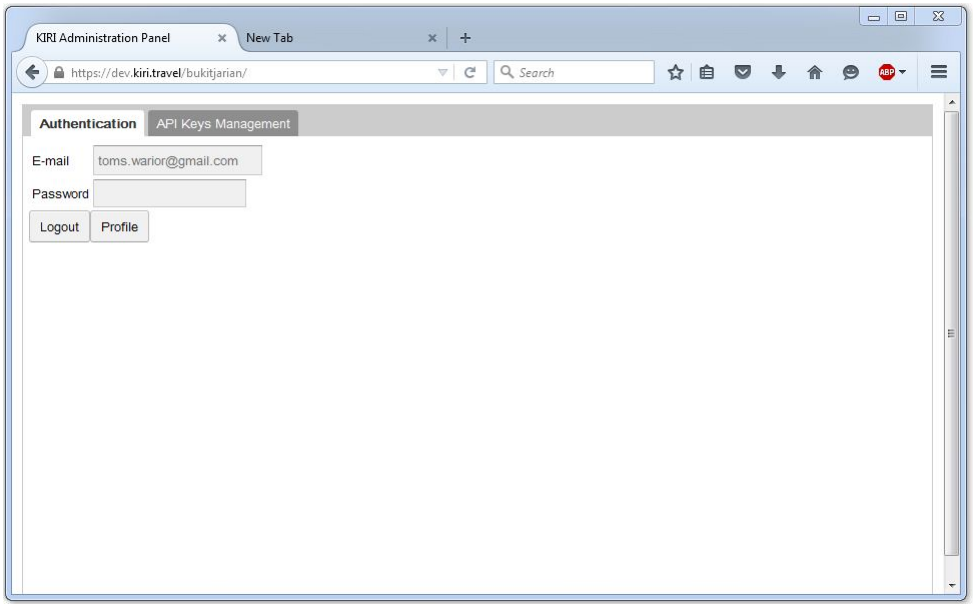


Gambar 1.1: Situs web KIRI[1]

KIRI *Dashboard*[2] (gambar 1.2) adalah bagian dari situs web KIRI. KIRI *Dashboard* berfungsi sebagai pengatur proses CRUD (*Create, Read, Update,* dan *Delete*) daftar rute yang terdapat dalam *database* situs web KIRI. KIRI *Dashboard Server Side* menggunakan bahasa PHP dalam pembuatannya[5]. Bahasa PHP kurang cocok untuk proyek skala besar seperti *dashboard*. Salah satu penyebab bahasa PHP kurang cocok adalah karena tidak ada deklarasi dan tipe variabel dalam penggunaan bahasa PHP.

Java merupakan bahasa pemrograman yang umum digunakan oleh banyak orang. Selain umum digunakan, Java juga merupakan bahasa pemrograman yang lebih terstruktur dibandingkan dengan PHP. Adanya deklarasi dan tipe variabel pada Java membuat setiap variabel memiliki kegunaan yang lebih jelas dan mudah dimengerti. Play Framework merupakan salah satu *framework* yang

membantu implementasi Java dalam pembuatan suatu situs web. Play Framework juga cocok untuk proyek skala besar karena arsitekturnya sudah menggunakan konsep MVC (*Model View Controller*)[3].



Gambar 1.2: KIRI *Dashboard*[2]

Berdasarkan ditemukannya kekurangan-kekurangan pada KIRI *Dashboard Server Side* seperti yang telah dijelaskan, maka solusi untuk mengatasi kekurangan tersebut adalah dibuatlah penelitian ini untuk mengubah KIRI *Dashboard Server Side* yang semula dalam bahasa PHP menjadi bahasa Java dengan menggunakan Play Framework.

1.2 Rumusan Masalah

Berikut adalah susunan permasalahan yang akan dibahas pada penelitian ini:

1. Bagaimana isi kode KIRI *Dashboard Server Side* dan apa saja kekurangan yang ada di dalamnya?
2. Bagaimana cara kerja Play Framework berbasis MVC?
3. Bagaimana melakukan *porting* KIRI *Dashboard Server Side* yang semula dalam bahasa PHP menjadi bahasa Java dengan menggunakan Play Framework?

1.3 Tujuan

Berdasarkan rumusan masalah yang telah dibuat, maka tujuan penelitian ini dijelaskan ke dalam poin-poin sebagai berikut:

1. Mengetahui isi kode KIRI *Dashboard Server Side* dan kekurangan-kekurangan yang ada di dalamnya.
2. Mengetahui cara kerja Play Framework berbasis MVC.

3. Melakukan *porting* KIRI *Dashboard Server Side* yang semula dalam bahasa PHP menjadi bahasa Java dengan menggunakan Play Framework.

1.4 Batasan Masalah

Penelitian ini dibuat berdasarkan batasan-batasan sebagai berikut:

1. Play Framework yang digunakan selama penelitian ini adalah versi 2.4.3.
2. *Porting* Kode KIRI *Dashboard Server Side* yang dilakukan adalah berdasarkan versi terbaru dari Github dengan *username*: "pascalalfadian"[5].

1.5 Metode Penelitian

Berikut adalah metode penelitian yang digunakan dalam penelitian ini:

1. Melakukan studi literatur mengenai kode KIRI *Dashboard Server Side*, MySQL Spatial Extensions, dan Play Framework.
2. Menganalisis teori-teori untuk membangun KIRI *Dashboard Server Side* dalam bahasa Java dengan menggunakan Play Framework.
3. Merancang KIRI *Dashboard Server Side* dalam bahasa Java dengan menggunakan Play Framework.
4. Melakukan *porting* kode situs web KIRI *Dashboard Server Side* menjadi Java dengan menggunakan Play Framework.
5. Melakukan pengujian terhadap fitur-fitur yang sudah dibuat.

1.6 Sistematika Penulisan

Setiap bab dalam penelitian ini memiliki sistematika penulisan yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan, yaitu membahas mengenai gambaran umum penelitian ini yang. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.
2. Bab 2: Dasar Teori, yaitu membahas mengenai teori-teori yang mendukung berjalannya penulisan ini. Berisi tentang MySQL Spatial Extensions dan Play Framework.
3. Bab 3: Analisis, yaitu membahas mengenai analisa masalah. Berisi tentang analisis kode KIRI *Dashboard Server Side* dan analisis kekurangan-kekurangan kode KIRI *Dashboard Server Side*.
4. Bab 4: Perancangan, yaitu membahas mengenai perancangan yang dilakukan sebelum melakukan tahapan implementasi. Berisi tentang perancangan fitur CRUD KIRI *Dashboard Server Side* menggunakan Play Framework, perancangan basis data, dan perancangan antarmuka KIRI *Dashboard* menggunakan Play Framework.

5. Bab 5: Implementasi dan Pengujian, yaitu membahas mengenai implementasi dan pengujian aplikasi yang telah dilakukan. Berisi tentang implementasi dan hasil pengujian aplikasi.

6. Bab 6: Kesimpulan dan Saran, yaitu membahas hasil kesimpulan dari keseluruhan penelitian ini dan saran-saran yang dapat diberikan untuk penelitian berikutnya. Berisi tentang kesimpulan dan saran.

BAB 2

DASAR TEORI

2.1 MySQL Spatial Extensions

Suatu *geographic feature*[6] adalah sesuatu yang ada di bumi yang memiliki lokasi sebagai penunjuk letak keberadaannya. Geometri adalah cabang ilmu matematika yang digunakan untuk memodelkan suatu *geographic feature*. Dengan geometri, suatu *geographic feature* dapat dinyatakan sebagai sebuah titik, garis, ruang, ataupun bentuk lainnya. Suatu "*feature*" yang dimaksud dalam istilah *geographic feature* dapat berupa:

1. ***An entity***, contohnya adalah gunung, kolam, kota, dll.
2. ***A space***, contohnya adalah daerah, cuaca, dll.
3. ***A definable location***, contohnya adalah persimpangan jalan, yaitu suatu tempat khusus dimana terdapat 2 buah jalan yang saling berpotongan.

MySQL adalah salah satu perangkat lunak yang digunakan untuk mengatur data-data (*database*) suatu situs web. Bentuk MySQL adalah sekumpulan tabel-tabel yang umumnya memiliki hubungan antar satu dengan yang lainnya. Setiap tabel pada MySQL memiliki kolom dan baris. Kolom pada MySQL menyatakan daftar jenis baris yang ingin dibuat dan baris menyatakan banyaknya data yang ada dalam tabel.

Penamaan suatu kolom dalam MySQL membutuhkan penentuan jenis tipe data yang akan digunakan dalam kolom tersebut. Dalam MySQL terdapat tipe-tipe data yang umum digunakan seperti *Varchar* untuk menyimpan karakter atau kata, *Int* untuk menyimpan angka, *Boolean* untuk menyimpan nilai "`true`" atau "`false`", dan tipe data lainnya. MySQL Spatial Extensions adalah perluasan dari tipe-tipe data yang disediakan MySQL untuk menyatakan nilai geometri dari suatu *geographic feature*.

2.1.1 Tipe Data Spatial

Berdasarkan kemampuan penyimpanan nilai geometri[7], tipe data *spatial* dapat dikelompokan ke dalam 2 jenis:

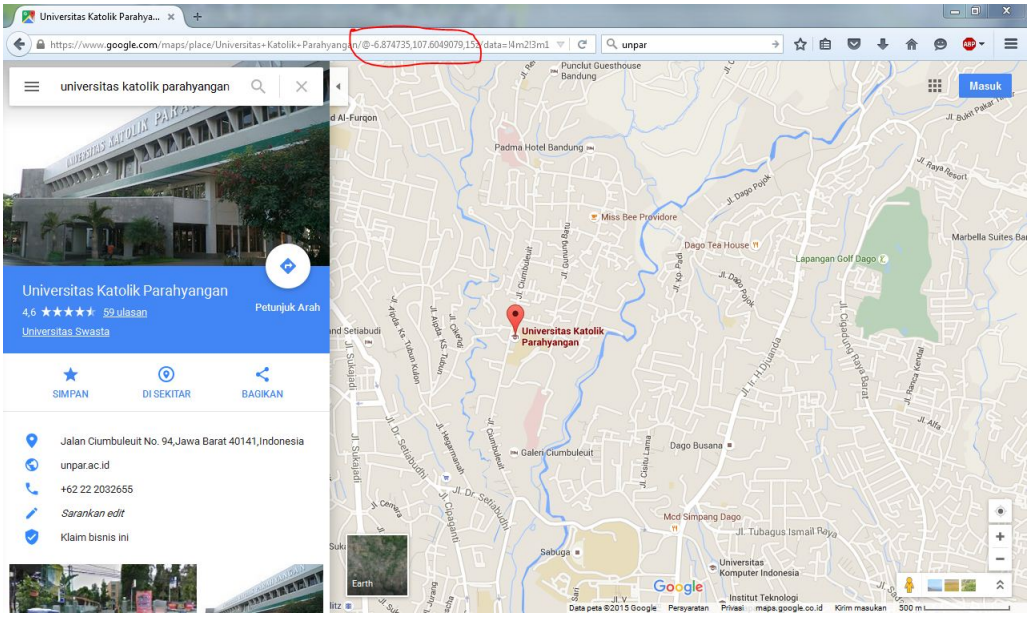
1. Tipe data yang hanya dapat menyimpan sebuah nilai geometri saja, yaitu:
   - ***Geometry***
   - ***Point***

- ***LineString***
- ***Polygon***

2. Tipe data yang dapat menyimpan sekumpulan nilai geometri, yaitu:
   - ***MultiPoint***
   - ***MultiLineString***
   - ***MultiPolygon***
   - ***GeometryCollection***

Point

Point adalah nilai geometri yang merepresentasikan sebuah lokasi ke dalam suatu koordinat[8]. Koordinat pada *Point* terdiri dari nilai X dan Y dimana X merepresentasikan letak lokasi dalam garis bujur dan Y merepresentasikan letak lokasi dalam garis lintang. *Point* tidak memiliki dimensi maupun nilai batasan. Contoh representasi *Point* adalah Universitas Katolik Parahyangan direpresentasikan dalam koordinat X=-6.874735 dan Y=107.6049079 pada skala tertentu (gambar 2.1).

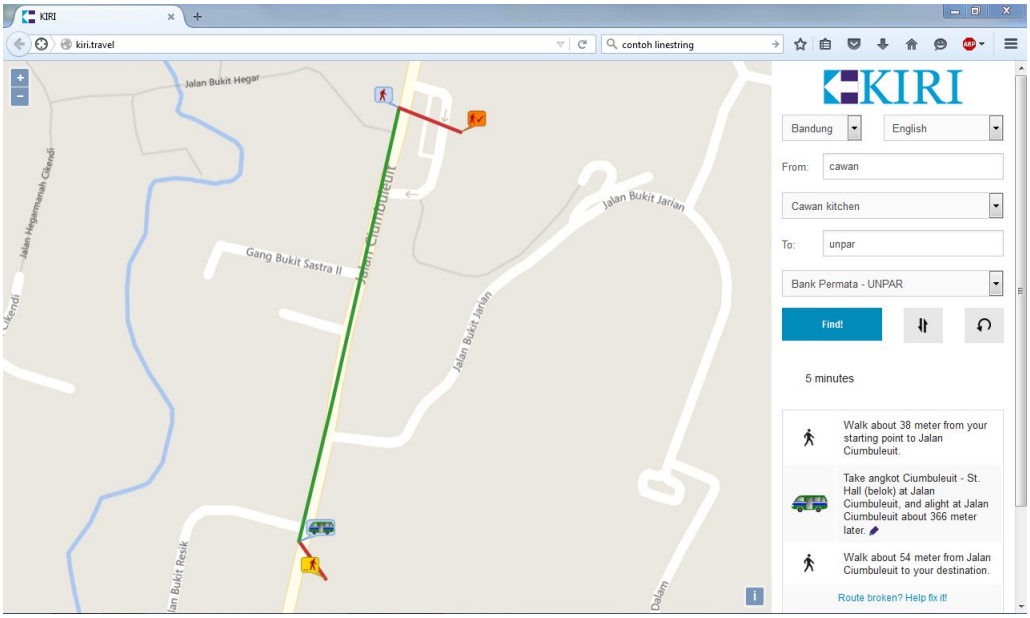


Gambar 2.1: Koordinat lokasi Universitas Katolik Parahyangan (**lingkaran merah**)

LineString

LineString adalah garis yang terbentuk dari sekumpulan *Point*[9]. Dalam peta dunia, *LineString* dapat merepresentasikan sebuah sungai dan dalam peta perkotaan, *LineString* dapat merepresentasikan sebuah jalan (contoh: gambar 2.2). Karena *LineString* merupakan sekumpulan *Point*, maka *LineString* menyimpan sekumpulan koordinat dimana setiap koordinat ($X_1..X_n$ dan $Y_1..Y_n$, dimana n menyatakan banyaknya *Point* dalam *LineString*) terhubung oleh garis dengan koordinat selanjutnya. Contohnya: misal terdapat sebuah *LineString* yang mengandung 3 buah *Point*, maka

terdapat garis yang menghubungkan *Point* pertama dengan *Point* kedua dan *Point* kedua dengan *Point* ketiga.



Gambar 2.2: Rute (jalan) yang harus ditempuh dari Cawan Kitchen untuk sampai ke Universitas Katolik Parahyangan direpresentasikan dengan *LineString* (**garis hijau** dan **garis merah**)

2.2 Play Framework

Play Framework adalah sekumpulan kerangka kode yang dapat digunakan untuk membangun suatu situs web. Play Framework tidak hanya menggunakan bahasa Java dalam pembuatannya. Bahasa Scala juga digunakan Play Framework dalam beberapa bagian seperti bagian *view* dan *route*[3]. Play Framework menggunakan konsep MVC (*Model View Controller*) sebagai pola arsitekturnya. Konsep MVC pada suatu kode membuat kode mudah dikembangkan baik secara tampilan maupun pengembangan fitur-fiturnya. Ketika *server* Play Framework dijalankan, secara *default* dapat diakses melalui "localhost:9000".

2.2.1 Struktur Aplikasi

Ketika Play Framework pertama kali ter-*install* pada komputer, Play Framework menyediakan *default* direktori dengan struktur minimal (gambar 2.3). Berikut adalah penjelasan struktur minimal Play Framework:

1. *Folder* "app" merupakan *folder* yang berisi mengenai pola arsitektur yang dimiliki Play Framework, yaitu "models" (tidak dibuat secara *default*), "views", dan "controllers" yang akan dijelaskan lebih lanjut pada subbab selanjutnya (subbab *Models*: 2.2.3, subbab *Views*: 2.2.4, dan subbab *Controllers*: 2.2.5).

2. *Folder* "conf" berisi mengenai *file* "application.conf" yang menyimpan pengaturan-pengaturan seperti kumpulan *log*, koneksi ke *database*, jenis *port* tempat *server* bekerja, dll. *Folder* "conf"

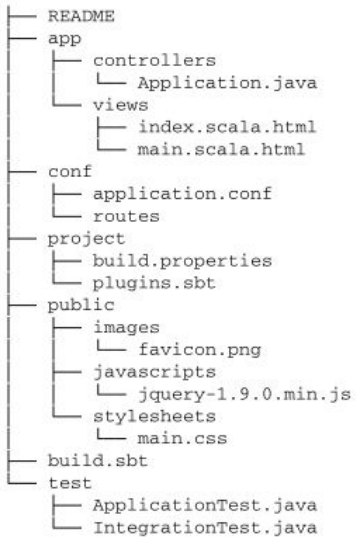
juga berisi *file* "routes" yang mengatur bagaimana HTTP *requests* nantinya akan diproses lebih lanjut yang akan dijelaskan pada subbab selanjutnya (subbab 2.2.2).

3. *Folder* "project" terdapat *file* "build.properties" dan "plugins.sbt", *file* tersebut mendeskripsikan versi Play dan SBT yang digunakan pada aplikasi.

4. *Folder* "public" merupakan *folder* yang menyimpan data-data seperti gambar (*folder* "images"), kumpulan Javascript yang digunakan (*folder* "javascripts", secara *default* berisikan *file* "jquery-1.9.0.min.js") dan data-data CSS (folder "stylesheets").

5. *File* "build.sbt" mengatur *dependencies* yang dibutuhkan dalam pembuatan aplikasi.

6. Terakhir adalah *folder* "test" yang merupakan salah satu kelebihan dari Play Framework, bagian ini berisikan *file* "Application.test" dan "Integration.test" yang dapat digunakan untuk melakukan serangkaian *testing* yang diinginkan terhadap aplikasi.



Gambar 2.3: Struktur minimal Play Framework

2.2.2 *Routes*

Routes adalah *file* yang mengatur pemetaan dari HTTP URLs menuju kode aplikasi (dalam hal ini menuju ke *controllers*). Secara *default*, *routes* berisikan kode seperti pada gambar 2.4. Kode *default* pada *routes* tersebut dapat memetakan permintaan URL *index* standar seperti "localhost:9000" ketika *server* Play Framework sudah dijalankan.

```
# Home page
GET     /   controllers.Application.index()

# Map static resources from the /public folder to the /assets URL path
GET     /assets/*file    controllers.Assets.at(path="/public", file)
```

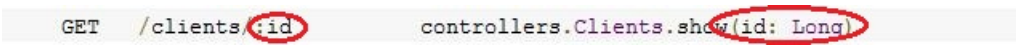
Gambar 2.4: Isi kode *file* "routes"[3]

Struktur *routes* terdiri dari 3 bagian (gambar 2.5), yaitu HTTP *method*, URL *path*, dan *action method*. Struktur *routes* seperti yang dijelaskan pada gambar 2.5 juga sekaligus menjadi struktur

minimal yang harus ada agar *routes* dapat memetakan suatu HTTP URLs. HTTP *method* berisikan protokol yang ingin dilakukan terhadap suatu HTTP *request*. HTTP *method* dapat berupa "`GET`", "`POST`", "`DELETE`", "`PATCH`", "`HEAD`" atau "`PUT`"[10]. URL *path* merupakan direktori yang ingin dituju dalam *server* aplikasi. URL *path* dimulai dengan tanda "/" dan diikuti dengan nama direktori yang ingin dituju. Terakhir, *action method* merupakan pemilihan kelas *controller* yang ingin dituju. Struktur *action method* terdiri dari 3 bagian (dipisahkan dengan karakter "."), yaitu pemilihan *package* "controllers" yang ingin dituju, bagian kedua adalah pemilihan kelas "controllers" yang dipilih (contohnya: "Products" pada gambar 2.5), dan terakhir adalah pemilihan *method* yang ada pada kelas "controllers" yang dipilih (contohnya: "list()").

Gambar 2.5: Struktur kode *file* "routes"[3]

URL *path* dan *action method* pada *routes* juga dapat berisi sebuah nilai variabel (gambar 2.6). Penulisan sebuah variabel pada URL *path* dimulai dengan tanda ":" lalu diikuti dengan nama variabel yang diinginkan, contohnya: "`:id`". Ketika menggunakan variabel pada URL *path*, pada *action method* perlu ditambahkan deklarasi variabel yang ditaruh di dalam bagian *method* yang dipilih. Cara penulisan deklarasi variabel pada *action method* adalah dimulai dengan nama variabel, lalu diikuti karakter ":", dan diakhiri dengan tipe variabel yang diinginkan. Contoh penulisan deklarasi variabel di dalam *method* suatu kelas pada bagian *action method*: "`id: Long`" seperti dijelaskan pada gambar 2.6.



Gambar 2.6: Struktur kode *file* "routes" dengan variabel[4] (**lingkaran merah**)

2.2.3 *Models*

Fungsi *models* pada Play Framework sama seperti fungsi *models* pada pola arsitektur MVC secara umum, yaitu untuk memanipulasi dan menyimpan data. Secara *default*, *models* tidak dibuat oleh struktur minimal Play Framework (gambar 2.3). Untuk itu perlu menambahkan *models* secara manual ke dalam struktur Play Framework. Langkah yang dilakukan untuk menambahkan *models* ke dalam Play Framework adalah:

1. Menambahkan folder "models" ke dalam folder "app",
2. Menambahkan file dengan format ".java" ke dalam folder "models".

Tidak ada aturan khusus yang diharuskan dalam penulisan kode dalam kelas *models*. Selama kelas *models* yang dibuat memenuhi aturan bahasa Java, maka *models* dapat dieksekusi oleh *server* Play Framework.

2.2.4 *Views*

Fungsi *views* pada Play Framework adalah mengatur tampilan yang ingin ditampilkan di layar. *Views* menggunakan bahasa HTML dan Scala. Bahasa Scala pada *views* berfungsi sebagai penerima parameter yang dikirimkan dari kelas *models* dimana antara *models* dan *views* dihubungkan oleh *controllers*. Penamaan *file* di dalam folder *views* (gambar 2.3) harus dengan format sebagai berikut, "namaFile.scala.html".

```
@(name:String)
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello</title>
  </head>
  <body>
    <h1>Hello <em>@name</em></h1>
  </body>
</html>
```

Gambar 2.7: Contoh struktur kode *views*[3]

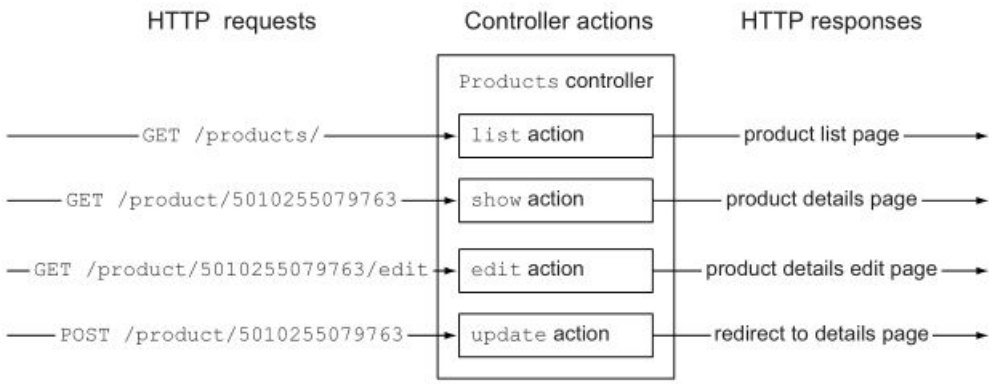
Baris pertama pada kode *views* (gambar 2.7) digunakan sebagai parameter penerima input dari *models* yang dihubungkan dengan *controllers*. Format deklarasi variabel pada parameter *views* diawali dengan karakter "@", lalu "(namaVariabel$_1$:tipeVariabel$_1$)(namaVariabel$_2$:tipeVariabel$_2$) . . . (namaVariabel$_n$:tipeVariabel$_n$)", dimana n adalah jumlah parameter yang ingin digunakan pada aplikasi *views*. Variabel pada parameter yang sudah dideklarikan dapat dipanggil dengan menggunakan format "@namaVariabel" (seperti dijelaskan pada baris 9 gambar 2.7).

2.2.5 *Controllers*

Controllers merupakan bagian pada Play Framework yang terhubung langsung dengan *routes* (subbab 2.2.2). Jika *action method* yang dikirimkan oleh *routes* sesuai dengan *method* yang dimiliki suatu kelas *controllers*, maka *controllers* akan mengeksekusi fungsi logika yang terdapat pada *method* dan mengembalikan nilai berupa objek dari kelas *Result* (gambar 2.8). Fungsi dari *controllers* dalam arsitektur MVC adalah sebagai penghubung antara *models* dan *views*.

Penulisan kode pada suatu kelas *controllers* menggunakan bahasa Java dan memiliki aturan khusus (gambar 2.9). Aturan khusus dijelaskan ke dalam poin-poin sebagai berikut:

1. Deklarasi kelas harus *public*,
2. Kelas yang dibuat harus *extends* "`play.mvc.Controller`",
3. *Method* yang dibuat dalam suatu kelas *controllers* harus bertipe *static*,
4. Nilai kembalian *method* yang dibuat dalam suatu kelas *controllers* harus berupa objek dari kelas Result.



Gambar 2.8: Contoh bagaimana hubungan *routes* dan *controllers* dalam memproses HTTP *requests*[3]

```
package controllers;
import play.mvc.Controller;
public class Products extends Controller {
}
```

Gambar 2.9: Contoh struktur kode *controllers*[3]

DAFTAR REFERENSI

[1] Pascal Alfadian, "KIRI." http://kiri.travel/, 2014. [Online; diakses 1-Oktober-2015].

[2] Pascal Alfadian, "KIRI." https://dev.kiri.travel/bukitjarian/, 2014. [Online; diakses 1-Oktober-2015].

[3] N. Leroux and S. D. Kaper, *Play for Java*. Manning Publications Co., 2014.

[4] Play Framework, "The routes file syntax." https://www.playframework.com/documentation/2.4.x/JavaRouting, 2015. [Online; diakses 12-Oktober-2015].

[5] Pascal Alfadian, "TirtayasaGH." https://github.com/pascalalfadian/TirtayasaGH, 2014. [Online; diakses 1-Oktober-2015].

[6] Oracle, "11.5 Extensions for Spatial Data." https://dev.mysql.com/doc/refman/5.0/en/spatial-extensions.html, 2015. [Online; diakses 1-Oktober-2015].

[7] Oracle, "11.5.1 Spatial Data Types." https://dev.mysql.com/doc/refman/5.0/en/spatial-datatypes.html, 2015. [Online; diakses 1-Oktober-2015].

[8] Oracle, "11.5.2.3 Point Class." https://dev.mysql.com/doc/refman/5.0/en/gis-class-point.html, 2015. [Online; diakses 1-Oktober-2015].

[9] Oracle, "11.5.2.5 LineString Class." https://dev.mysql.com/doc/refman/5.0/en/gis-class-linestring.html, 2015. [Online; diakses 1-Oktober-2015].

[10] Play Framework, "The HTTP method." https://www.playframework.com/documentation/2.4.x/JavaRouting, 2015. [Online; diakses 12-Oktober-2015].

LAMPIRAN A

THE SOURCE CODE

Listing A.1: MyFurSet.java

```
1
2  import java.util.ArrayList;
3  import java.util.Collections;
4  import java.util.HashSet;
5
6  /**
7   *
8   * @author Lionov
9   */
10
11 //class for set of vertices close to furthest edge
12 public class MyFurSet {
13     protected int id;                                    //id of the set
14     protected MyEdge FurthestEdge;                       //the furthest edge
15     protected HashSet<MyVertex> set;                     //set of vertices close to furthest edge
16     protected ArrayList<ArrayList<Integer>> ordered;     //list of all vertices in the set for each
           trajectory
17     protected ArrayList<Integer> closeID;                //store the ID of all vertices
18     protected ArrayList<Double> closeDist;               //store the distance of all vertices
19     protected int totaltrj;                              //total trajectories in the set
20
21     /**
22      * Constructor
23      * @param id : id of the set
24      * @param totaltrj : total number of trajectories in the set
25      * @param FurthestEdge : the furthest edge
26      */
27     public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
28         this.id = id;
29         this.totaltrj = totaltrj;
30         this.FurthestEdge = FurthestEdge;
31         set = new HashSet<MyVertex>();
32         ordered = new ArrayList<ArrayList<Integer>>();
33         for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
34         closeID = new ArrayList<Integer>(totaltrj);
35         closeDist = new ArrayList<Double>(totaltrj);
36         for (int i = 0;i <totaltrj;i++) {
37             closeID.add(-1);
38             closeDist.add(Double.MAX_VALUE);
39         }
40     }
41
42     /**
43      * set a vertex into the set
44      * @param v : vertex to be added to the set
45      */
46     public void add(MyVertex v) {
47         set.add(v);
48     }
49
50     /**
51      * check whether vertex v is a member of the set
52      * @param v : vertex to be checked
53      * @return true if v is a member of the set, false otherwise
54      */
55     public boolean contains(MyVertex v) {
56         return this.set.contains(v);
57     }
58 }
```