Training Java Fundamental dan Java Web

Ifnu Bima – Development Manager blibli.com
Kerjasama blibli.com dan ILKOM IPB

Instructor : Ifnu Bima

- 11 Tahun coding java
- IPB ilmu computer angkatan 38 (2001)
- UX Team Development Manager di blibli.com
  - UI blibli.com : Desktop Web dan Mobile Web
  - Native Mobile Apps : Android, iOS dan (soon) Windows Phone
  - CMS : Banner, Dynamic Pages, SEO Setting dll
  - Product Discovery : Search, Catalog, Product Review, Wishlist, Recommendation dll
- Ifnu.b.Fatkhan@gdn-commerce.com

Perkenalan

- Nama :
- Instansi / Kantor :
- Posisi :
- Sudah mengenal Java?
- Sudah mengenal pemrograman lain?
- Apa yang diharapkan setelah selesai training?

Materi

- Hari 1
  - Apa itu Java
  - Membuat aplikasi sederhana
  - Syntax Java
    - Keywords
    - Identifiers
    - Access Control
  - Control dan Iteration
    - If-else
    - Switch
    - For
    - while
  - Class dan Object
  - Data Type

Materi

- Hari 1 : Bonus
  - NetBeans
  - Maven
  - Unit Testing menggunakan JUnit

Materi

- Hari 2
  - OOP
    - Enkapsulasi (Encapsulation)
    - Turunan (Inherritance)
    - Polimorfisme (Polymorphism)
    - Overriding dan Overloading
    - Casting variabel reference
    - Interface
  - Collections dan Generics
    - Collection
    - List
    - Set
    - Queue
    - Map
  - Feature Java 5 dan Java 7
  - Java 8, functional programming

Materi

- Hari 2 : Bonus
  - JDBC
    - Connection
    - Statement
    - ResultSet
    - PreparedStatement
    - Batch Execution
  - Entity/Model
  - Transaction
  - DAO Pattern
  - Service Pattern dan Transaction management

Apa itu Java?

- Java platform
- Sejarah java
- Java ME, Java SE, Java EE
- JVM
- Industri yang menggunakan Java

Praktek

- HelloWorld
- Java Command Line
- Write – Compile - Execute

Environment Setting

1. klik kanan my computer, pilih properties
2. setelah terbuka jendela properties, pilih tab advance
3. di dalam path path advance klik tombol system variables
4. di dalam jendela system variables pilih baris yang terdapat path, klik tombol edit
5. tambahkan folder C:\Program Files\Java\1.6.0_23\bin diakhir dari pathnya, jangan lupa menambahkan ; sebagai pemisah
6. test langkah-langkah di atas dengan menjalankan perintah berikut ini dari command prompt :

```
$ javac -version
javac 1.6.0_22
```

Syntax Java

- Diturunkan dari C/C++

Keyword

abstract boolean break byte case catch
char class const continue default do
double else extends final finally float
for goto If implements import instanceof
int interface long native new package
private protected public return short
static
strictfp super switch synchronized this
throwthrows transient try void volatile
While assert enum

Identifiers

- Identifers adalah nama-nama yang bisa dideklarasikan dalam java tetapi bukan bagian keyword
- identifers antara lain: class, interface, variabel/property dan method
- Aturanya :
  - Aturan pertama sudah kita bahas sebelumnya adalah semua keyword java tidak boleh digunakan sebagai identifers.
  - Identifers harus diawali oleh huruf, simbol mata uang dolar($) atau karakter penghubung
  - underscore (_). Angka tidak boleh digunakan sebagai karakter pertama identifers.
  - Setelah karakter pertama, berikutnya boleh diikuti oleh huruf, simbol mata uang dolar,
  - karakter penghubung, dan angka.
  - Tidak ada pembatasan panjang identifers
  - Identifers di java bersifat case-sensitif, foo dengan Foo adalah dua buah identifers berbeda.
  - Nama public class harus sama persis dengan nama fle .java

Contoh Identifiers

- Contoh identifier yang valid :
  - `int x;`
  - `int $123;`
  - `int ini_nama_identifiers_yang_panjang_dan_alay;`
- Identifier yang tidak valid :
  - `int _x;`
  - `int 1y;`
  - `int ______17_r;`
  - `int _$;`

Access Modifier

- `public` : menandakan bisa diakses oleh siapapun tanpa batasan.
- `protected` : bisa diakses oleh class turunanya dan class-class lain yang berada dalam package yang sama
- **default** tidak memerlukan keyword, kalau tidak ada salah satu dari tiga access modifer lain maka yang digunakan adalah access modifer **default**.
  - class yang sama
  - class lain dari package yang sama dengan class tersebut
- `private` : hanya mengijinkan diakses oleh class yang sama

Statement

- Satu baris (atau lebih) kode java
- Satu bagian kode yang bisa berdiri sendiri
- Di akhiri dengan simbol ;
- Statement Block
  - Terdiri dari banyak statement
  - Kurung kurawal buka ({) menandai awal dari blok
  - Kurung kurawal tutup (}) menandai akhir dari blok

Flow Control

- if

```
if(x == 1) {
  System.out.println("nilai X adalah 1");
}
```

- if-else

```
if(x == 1) {
  System.out.println("nilai X adalah 1");
} else {
  System.out.println("nilai X bukan 1");
}
```

- if-else if-else

```
if(x == 1) {
  System.out.println("nilai X adalah 1");
} else if(x<0) {
  System.out.println("nilai X adalah negatif");
} else {
  System.out.println("nilai X bukan 1 dan bukan negatif");
}
```

Flow Control

Switch

```
switch(x){
   case 1 :
      System.out.println("satu");
      break;
   case 2 :
      System.out.println("dua");
      break;
   default :
      System.out.println(
         "bukan satu atau dua");
}
```

Iterasi / Perulangan

- for

```
for(initial condition;stop
conditon;iteration)
```

- while

```
while(condition)
```

- do-while

```
do{
} while(condition)
```

Anatomi Class Java

- Deklarasi class
- Method
- Constructor
- Property
- Konstanta

Anatomi Class Java

```
public class Person {   //deklarasi class

  //konstanta
  public static final String GENDER_MALE = "male";
  public static final String GENDER_FEMALE = "female";

  private String name; //property

  public Person(){    //constructor
  }

  public void setName(String personName){    //method
    name = personName;
  }
}
```

Class dan Object

- Object is instantiation of a class

```
public class Person{
     private String name;
     public void setName(
          String personName){
          name = personName;
     }
}

Person fulan = new Person();
fulan.setName("Fulan Wijaya");
```

Data Type

- Primitive
- Wrapper
- Object reference

Primitive dan Passangan Wrapper

- int, Integer
- long, Long
- float, Float
- double, Double
- byte, Byte
- short, Short
- booelan, Boolean

Wrapper

- String
- Date
- Timestamp
- BigDecimal
- BigInteger

Object Reference

```
Person fulan = new Person();
```

fulan adalah object reference

Array

- Array adalah object di java yang dapat menyimpan kumpulan data dengan tipe yang sama.
- Array dapat menyimpan data dengan tipe primitive, wrapper maupun object reference
- Array adalah Object
- Untuk bisa bekerja dengan array, kita perlu melakukan tiga langkah :
  - Mendeklarasikan variabel array

```
int[] arrayInteger;
int arrayInteger[];
```

  - Menginstansiasi object array

```
arrayInteger  = new int[10];
```

  - Mengisi array dengan data

```
for(int i=0; i < arrayInteger.length; i++){
  arrayInteger[i] = i * 100;
}
String[] days = new String[]{"senin","selasa"}
String[] days = {"senin", "salasa"}
```

NetBeans

- IDE untuk memudahkan development
- Feature yang sering digunakan
  - Build otomatis untuk mengetahui ada error dalam kode
  - Run dan Debug
  - Syntax highlighting
  - Otomasi

Maven

- Build Tools dan Dependency Management
- Maven Repository
  - Kumpulan library: memudahkan menambahkan library tanpa harus download manual
- Build Tools lain :
  - Ant
  - Ivy
  - Gradle
- Menghilangkan ketergantungan dengan IDE
- Otomasi

Unit Testing : JUnit

- Mengetes logic aplikasi secara otomatis dan berulang ulang
- Menghindarkan developer mengetes dengan manual menggunkan mata
- Menghindari “regression bug”: mengubah satu logic / kode membuat logic / kode lain jadi error (safety net)
- Membuat developer percaya diri untuk melakukan refactoring karena ada safety net dari Unit Testing
- Digunakan secara luas sebagai ukuran internal code quality
- Junit secara de facto diterima sebagai tools untuk Unit Testing

Materi Hari 2

- OOP
  - Enkapsulasi (Encapsulation)
  - Turunan (Inherritance)
  - Polimorfisme (Polymorphism)
  - Overriding dan Overloading
  - Casting variabel reference
  - Interface
- Collections dan Generics
  - Collection
  - List
  - Set
  - Queue
  - Map

OOP : Enkapsulasi

- Arti harfiah : pembungkusan
- Apa yang dibungkus?
  - Karakteristik (*state*)
  - Perilaku (*behavior*)
- Dalam Java, Class adalah perwujudan dari konsep enkapsulasi karakteristik dan perilaku dalam satu tempat
  - Karakteristik diwujudkan sebagai *property*
  - Perilaku diwujudkan dalam sebuah *method*
- Procedural programming language seperti C memisahkan *state* dan *behaviour* di tempat berbeda

OOP : Enkapsulasi

- Enkapsulasi vs *Access Modifier*
  - `public`
  - `protected`
  - *default*
  - `private`
- *Access Modifier* bagian dari enkapsulasi untuk mengontrol **siapa** bisa mengkakses **apa**

OOP : Enkapsulasi

```
public class Human {
  private String name; //karakteristik

  public void shout(){ //perilaku
    System.out.println(
      "HEY!! I'm " + name + "!!");
  }
}
```

OOP : Turunan

- Penurunan karakteristik dan perilaku dari orang tua (parent) ke anaknya (child)
- Yang diturunkan harus ditandai dengan *access modifier* :
  - `public`
  - `protected`
- Penurunan bisa dianggap sebagai hubungan *IS-A*
- Semua class dalam java adalah turunan dari class `Object`
- Turunan dalam java menggunakan keyword `extends`

OOP : Turunan, hubungan *IS-A*

```
public class Customer {}

public classs MemberCustomer
extends Customer {}
```

- `MemberCustomer` **is a** `Customer`
- `MemberCustomer` **is a** `Object`
- `Customer` **is a** `Object`

OOP : Polimorfisme

- Arti harfiah : Banyak (poly) Bentuk (morph)
- Apa yang banyak bentuknya?
  - *class and object* mempunyai banyak bentuk
  - *method* mempunyai banyak bentuk
- Class mempunyai banyak bentuk karena turunan
  - `MemberCustomer` is a `Object`, karena semua *class* pasti `extends Object` secara implisit
  - `MemberCustomer` is a `Customer`, karena `MemberCustomer extends Customer`
  - `MemberCustomer` is a `MemberCustomer`

OOP : Polimorfisme

- Object mempunyai banyak bentuk karena turunan

```
Object objectX = new MemberCustomer();
Customer customerX = new MemberCustomer();
MemberCustomer memberX = new MemberCustomer();
Object objectY = memberX;
customerX = memberX;
```

OOP : Polimorfisme pada method Overloading

- Method dibedakan berdasarkan :
  - Nama
  - Tipe parameter
  - Jumlah parameter
  - Posisi parameter
- Overloading : method yang namanya sama dengan parameter berbeda. Baik dari tipenya, dari banyaknya atau dari posisinya. Contoh :

```
public void shout(){
    System.out.println("Hey !! I'm " + name + "!!");
}

public void shout(String words){
    System.out.println(name + " shouting " + words + "!!!");
}

public void shout(int times){
    System.out.println(name + " shouting " + times + " times !!!");
}

public void shout(String words, int times){
    System.out.println(name + " shouting " + words
        + "  " + times + " times !!!");
}
```

OOP : Polimorfisme pada method Overriding

- Overriding method dilakukan dengan cara mendeklarasikan method yang sama persis dengan method yang ada di class orang tuanya.
- Override method toString dari class `Object`

```
public String toString(){
   return “to string method overridden”;
}
super.toString();
```

- Menambahkan annotation `@Override` untuk memastikan aturan overriding dipatuhi.

Casting object reference

- Memaksa object dari orang tua (parent) menjadi tipe anaknya (child)

```
Customer customerX = new MemberCustomer();
MemberCustomer memberX = (MemberCustomer) customerX;
```

- Gunakan *keyword* `instanceof` dalam *if statement* untuk mengetest *object* sebelum melakukan *casting*

```
Customer customerX = new MemberCustomer();
if(customerX instanceof MemberCustomer){
  MemberCustomer memberX = (MemberCustomer) customerX;
}
```

- Memaksa object apapun menjadi tipe lain bisa mengakibatkan *class cast exception*

```
String customerX = "customer with name of X";
MemberCustomer memberX = (MemberCustomer) customerX;

Customer customerX = new Customer();
MemberCustomer memberX = (MemberCustomer) customerX;
```

Interface

- Class spesial yang semua methodnya hanya berupa deklarasi tanpa implementasi
  - Semua *method* otomatis mempunyai atribut `public` dan `abstract`
  - Semua property otomatis mempunyai atribut `public`, `static` dan `final`
- Menggunakan keyword `interface` dalam deklarasi dan `implements` dalam penggunaan
- *Program to Interface* dianggap sebagai praktek yang baik

Interface

```
public interface Shouter {
   void shout();
}

public class Human implements Shouter{
   private String name;

   @Override
   public void shout(){
      System.out.println(
         "HEY!! I'm " + name + "!!");
   }
}
```

Penggunaan Interface

```
Human rijen = new Human();
Shouter shouter = rijen;

Shouter rijenShouter = new Human();
```

Java Collection Framework

- *Interface* `Collection`
- *Interface* `Set`
  - *Class* `HashSet`
- *Interface* `SortedSet`
  - *Class* `TreeSet`
- *Interface* `List`
  - *Class* `ArrayList`
- *Interface* `Map`
  - *Class* `HashMap`
- *Interface* `SortedMap`
  - *Class* `TreeMap`
- `Queue`
- `NavigableSet`
- `NavigableMap`

List

- `List` adalah jenis `collection` yang teratur tetapi tidak terurut.
- `List` mempunyai index yang disusun berdasarkan urutan kapan item dimasukkan ke dalam List
- Isi dari `List` bersifat tidak unik, alias dua buah *item* yang sama bisa dimasukkan berkali kali ke dalam `List`
- Method dalam `List`
  - `get(int index)` *method* ini digunakan untuk mengambil isi dari list berdasarkan index
  - `indexOf(Object o)` *method* ini digunakan untuk mengetahui berapa nomor index dari *object* yang ada dalam `List`.
  - `add(Object o)` *method* digunakan untuk menambahkan *object* ke dalam `List`
  - `add(int index, Object o)` menambahkan *object* ke dalam `List` di *index* tertentu

Contoh List

- Potong kalimat dan urutkan berdasarkan kata
- Input : “Saya sedang training java di ilkom ipb”
- Output :
  - di
  - ilkom
  - ipb
  - java
  - Saya
  - sedang

Set

- `Set` adalah `collection` yang bersifat unik
- Defnisi unik diimplementasikan dengan mengoverride *method* `equals` dan `hashCode` dari *class* yang akan dimasukkan ke dalam `Set`
- *Object* yang dimasukkan ke dalam Set harus memperhatikan konsep "*Object Equality*"

Object Equality

- `==` vs `equals`

```
int i = 10;
i == 10; //true

String str1 = "Apa aja";
String str2 = str1;
str1 == str2;  //true
str1.equals(str2); //true
```

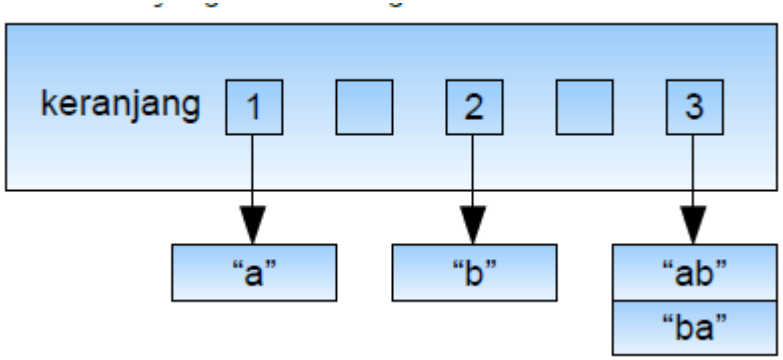
- `==` bernilai `true` kalau "alamat *memory*" dua *object* sama
- `equals` bernilai `true` kalau secara "*logic*" dua *object* sama
  - `equals` adalah method yang diturunkan dari class `Object`
  - Kalau tidak di*override*, *logic* di dalam method `equals` menggunakan `==`
  - Dalam contoh di atas, *class* `String` meng*override* method `equals` agar dua *object* `String` dianggap sama kalau susunan hurufnya sama
- Method `hashCode` mengembalikan nilai numerik dari suatu *object*

Aturan mengoverride `equals` dan `hashCode`

- Method `hashCode` harus mengembalikan nilai yang sama walaupun dieksekusi berkali-kali selama nilai *property* dalam *object* tidak berubah.
- `a.equals(b)` return `true`: *method* `hashCode` dari kedua *object* `a` dan `b` **harus** mengembalikan nilai *integer* yang **sama**.
- **Sebaiknya**, `a.equals(b)` return `false` : *method* `hashCode` dari kedua *object* `a` dan `b` mengembalikan nilai integer yang **berbeda**.
- `a.equals(b)` return `false` : *method* `hashCode` dari kedua *object* `a` dan `b` **tidak harus** mengembalikan nilai yang **berbeda**, **boleh** mengembalikan nilai yang **sama**.

Set

- Visualisasi struktur data `Set`



- Ke kanan (index) menggunakan nilai `hashCode`. Kalau keranjang mempunyai banyak item, ke bawah menggunakan `equals`
- Implementasi `hashCode` yang baik memastikan pencarian (method `contains`) dari `Set` mempunyai kompleksitas algoritma **O(1)**

Map

- `Map` adalah bentuk struktur data yang berpasangan antara *key-value*.
- Key bersifat unik karena implementasinya menggunakan `Set`

Java Database Connectivity

- JDBC
  - Connection
  - Statement
  - ResultSet
  - PreparedStatement
  - Batch Execution
- Entity/Model
- Transaction
- Design Pattern
  - DAO Pattern
  - Service Pattern dan Transaction management

Java Database Connectivity

- MySQL JDBC Driver
- Connection
  - connect ke MySQL server
- Statement
  - menjalankan query
- ResultSet
  - menampung hasil query
- PreparedStatement :
  - mengcompile query terlebih dahulu
  - menambahkan parameter kemudian
  - menghindari SQL injection
- Batch Execution
  - eksekusi query ditampung di client kemudian dikirim ke server secara bersamaan

Design Pattern

- Model Pattern
- Model vs DTO pattern
- Data Access Object (DAO) pattern
- Service Pattern

Model Pattern

- Mapping tabel database ke dalam class
  - ORM : Object Relational Mapping
- Memudahkan membaca kode dengan mengaitkan nama class dengan konsep nyata
- Membuat aplikasi lebih terlihat OOP

DAO Pattern

- Prinsip separation of concern
  - Pemisahan fungsi yang berbeda diletakkan dalam class yang berbeda
  - Semua query dilakukan di dalam DAO
- Code to interface
  - Abstraksi proses akses data ke database
  - Membuat interface untuk DAO
  - Kalau pindah database tinggal implement dao berbeda

Service Pattern

- Prinsip separation of concern
- Memisahkan kode untuk menangani transaction
- Bussiness process diletakkan di service

Database Unit Testing dengan DBUnit

- Extensi JUnit untuk aplikasi yang menggunakan database
- Menyiapkan database ke keadaan tertentu sebelum test dijalankan
- DDL dan DML (bisa dari XML data)
- Export import data ke XML agar bisa digunakan di database berbeda

Object Relational Mapping (ORM)

- Framework menjembatani perbedaan sistem basis data yang bersifat relational dengan paradigma pengembangan aplikasi yang berorientasi objek
- Generate spesifik sql query setiap database
- Hibernate de facto ORM di java

Java Persistence API

- Interface untuk menjembatani banyak implementasi ORM
- Standard untuk mapping class java dan table database
- Generic query : JPA QL
- Hibernate Entity Manager adalah implementasi JPA dari Hibernate ORM

Mapping JPA sederhana

```
@Entity
@Table(name = "CUSTOMER")
public class Customer {

    @Id
    @Column(name="ID")
    private Integer id;

    @Column(name="NAME")
    private String name;

    @Column(name="EMAIL")
    private String email;

    @Column(name="ADDRESS")
    private String address;

    @Column(name="BIRTH_DATE")
    @Temporal(TemporalType.DATE)
    private Date birthDate;

    //getter dan setter
}
```

JPA QL

- Generic query di JPA, menggunakan nama class sebagai query
- JPA QL hanya ada untuk select saja, contoh:
  - `select c from Customer c`
  - `select c from Customer c where c.id=:id`
- Tidak perlu query untuk insert, update dan delete karena degenerate pada waktu aplikasi berjalan
- Generate native query ke database sesuai dengan Dialect yang digunakan, misalnya :
  - MySQL Dialect
  - Oracle Dialect
  - PostgreSQL Dialect
- Syntax lebih sederhana dan lebih pendek dibanding SQL query

Spring Data JPA

- Salah satu bagian / komponen dari Spring Data
- Mengurangi "*boiler plate code*" cukup signifikan dengan menambahkan
- Menggunakan repository design pattern, tidak menggunakan DAO design pattern
- Method dalam interface repository menjadi sangat pendek

Spring Data JPA

```
public interface CustomerRepository extends
    JpaRepository<Customer, Integer> {


}
```

- Hanya dengan deklarasi interface repository di atas, sudah tersedia secara otomatis method-method di bawah ini tanpa perlu coding sama sekali:
  - save
  - delete
  - findOne
  - findAll
  - findAll(Pageable page)

Java Web

- Salah satu profile dari Java EE
- Teknologi dalam Java Web Profile yang termasuk dalam training :
  - Servlet
  - JSP
  - JPA
- Membuat aplikasi web dengan tampilan di browser dan diakses oleh user
- Membuat aplikasi RestFull sebagai API untuk diakses aplikasi lain, misalnya
  - Integrasi dengan aplikasi lain
  - Mobile Apps : Android, windows, iOS

Java Web

- Package aplikasi web adalah war, singkatan dari web archive. File war pada dasarnya adalah file zip, jadi bisa dibuka menggunakan aplikasi winzip, 7zip atau winrar.
- War dideploy ke web server, misalnya tomcat atau glassfish
- Di dalam war ada beberapa jenis file
  - File class hasil compile aplikasi
  - File jar library yang diperlukan aplikasi
  - File JSP untuk mengenerate HTML
  - Static file seperti JS, CSS maupun HTML
  - File konfigurasi seperti web.xml, dll

Tomcat

- Tomcat adalah Java Web Server
- Secara de facto adalah web server paling popular di java
- Cukup ringan dengan memory footprint yang tidak terlalu besar
- Gratis dan open source
- Versi terakhir adalah versi 8

Servlet

- Servlet adalah class java special yang digunakan untuk menghandle URL dan memproses request dari user
- Servlet akan mengembalikan response bermacam-macam, tergantung logic aplikasi.
- Servlet dimapping dalam web.xml

Workshop

- Servlet HelloWorld
- Servlet mengenerate HTML

JSP

- Scripting language
- Membuat aplikasi java web dengan menambahkan kode java ke dalam HTML, mirip PHP.
- Pada waktu aplikasi berjalan, JSP akan decompile menjadi Servlet

MVC Pattern

- Membuat aplikasi web hanya menggunakan Servlet sangat tidak efektif dan efisien
- Membuat aplikasi web hanya menggunakan JSP akan membuat kode susah dibaca, strukturnya campur aduk dalam file JSP.
- MVC Pattern menggabungkan JSP dan Servlet
  - Model : representasi data dari datasource, misalnya database
  - View : JSP untuk membuat HTML saja
  - Controller : Servlet untuk logic menghandle URL dan memanggil kode untuk akses datasource

Spring Framework

- De facto framework untuk pengembangan aplikasi java
- Bersaing dengan standard Java EE
- Terdiri dari banyak project yang memudahkan development aplikasi java, apapun arsitekturnya
- Keuntungan utama menggunakan Spring :
  - Dependency Injection (DI) menyederhanakan struktur aplikasi
  - Mengurangi jumlah kode yang ditulis
  - Aspect Oriented Programming (AOP) framework
  - Spring Framework Project yang sangat banyak
  - Gratis dan open source
  - Profesional support dari Pivotal
- Dokumentasi dan tutorial sangat banyak, salah satu topik popular di stackoverflow

Spring MVC

- Implementasi MVC pattern dari Spring Framework
- Support pengembangan aplikasi web maupun aplikasi RestFull API
- Dokumentasi dan tutorial sangat banyak
- De Facto MVC framework di Java
- Support Spring Security, Spring Security Oauth
- Spring MVC test framework memudahkan unit testing untuk Controller

THANK YOU