

@khannedy

[illegible]

Daftar Isi

Daftar Isi	2
Mau ngapain di buku ini?	3
Tujuan bikin RESTful Web Service?	3
Dibuku ini kita mo bikin apa?	4
Cukup!	4
Bikin project-nya dulu yuk!	5
Bootstrap project.....	5
Buka project-nya pake apa?.....	5
Ngejalanin projectnya	7
Nyiapin data dulu.....	9
Bikin kelas model barang-nya	9
Bikin kelas status.....	10
Bikin RESTful Web Service-nya	12
Bikin kelas BarangController-nya	12
Nyiapin controller-nya dulu.....	12
Nambah data barang	13
Ngambil data barang	15
Ngubah data barang.....	16
Ngapus data barang	17
Ngambil semua data barang	18
Ngetest di client.....	19
Pake POSTMAN	19
Ngetest nambah barang.....	19
Ngetest ngambil barang.....	20
Ngetest ngubah barang	21
Ngetest ngapus barang.....	22
Ngetest ngambil semua data barang.....	23
Selesai!	24
Apa nih yang belum selesai?	24
Nambahin autentikasi pake BASIC AUTH.....	24
Ngetest lagi yang pake BASIC AUTH	25
Selesai nih?	27
Download kodenya dimana?	27
Buku siapa nih?	28

Mau ngapain di buku ini?

Ada yang bingung dengan judul buku ini? Kalo bingung ngapain juga dibaca! :P

Khusus buat yang gak ngerti dengan judul buku ini, saya jelasin sedikit, tentang apa itu RESTful Web Service.

RESTful web service adalah... wah males juga nih jelasinnya, udah liat aja deh di Wikipedia :P

http://en.wikipedia.org/wiki/Representational_state_transfer

Baca dulu ya, jangan maen praktekin aja nih buku, tar bisa GALAU tingkat DEWA lagi.

Tujuan bikin RESTful Web Service?

Apa sih tujuannya bikin RESTful web service? Sebenarnya sederhana, RESTful web service itu digunakan untuk membuat aplikasi berbasis client server, bedanya aplikasi client server nya itu basisnya web menggunakan HTTP protokol, jadi lebih mudah dan gampang bikinnya dibandingkan pake SOCKET (jangan nanya apa itu socket ya :P) atau TCP/IP

Karena RESTful web service itu menggunakan protocol HTTP, jadi enak nya, untuk melakukan testing bisa pake browser aja, tanpa harus bikin aplikasi client nya dulu.

Biasanya kapan sih dipake RESTful web service?

Kebanyakan sih sebenarnya RESTful web service itu dipake buat client aplikasi mobile.

Jadi kalo bukan mobile gak bisa?

Enggak juga, semua bisa!!!! Cuma karena memang proses RESTful web service itu sederhana (hanya menggunakan HTTP), jadi gampang sekali diimplementasikan, khususnya di mobile. Bahkan sangking sederhananya, sekarang hampir semua web service pake RESTful, udah jarang yang pake SOAP lagi

Apaan tuh SOAP?

Beud dah, nanya mulu, buka wikipedia sana!

<http://en.wikipedia.org/wiki/SOAP>

Dibuku ini kita mo bikin apa?

Di buku ini kita gak akan bikin RESTful web service yang aneh2, cukup bikin RESTful web service untuk CRUD (Create Read Update Delete) aja, simple, yang penting ngerti konsep nya, dan gimana cara bikinnya.

Dibuku ini kita bakal pake framework Java yang sangat populer, yaitu Spring Framework, khususnya teknologi web nya, Spring Web MVC.

Sengaja gw pake Spring Web MVC, jadi kalo lo dah ngerti nih teknologi, gak rugi deh, soalnya banyak perusahaan yang pake Spring Framework.

Cukup!

Cukup deh ye, basa basinya, kita langsung praktek aja, biar gak ngantuk Kemon!

Bikin project-nya dulu yuk!

Bootstrap project

Bootstrap (bikin project dari nol) project emang agak cape, soalnya kita banyak konfigurasi ini itu, sebelum mulai bener2 coding.

Maka dari itu, gw udah siapin project template, jadi lo gak usah ribet2 buat bootstrap project.

Project template nya bisa di download disini :

<https://github.com/khannedy/spring-mvc-rest-template/archive/master.zip>

Atau kalo ngerti GIT version control, silahkan clone disini :

git clone <https://github.com/khannedy/spring-mvc-rest-template.git>

Buka project-nya pake apa?

Gw bikin projectnya pake Apache Maven, ...

Kang, gw gak ngerti Apache Maven, baru denger malah

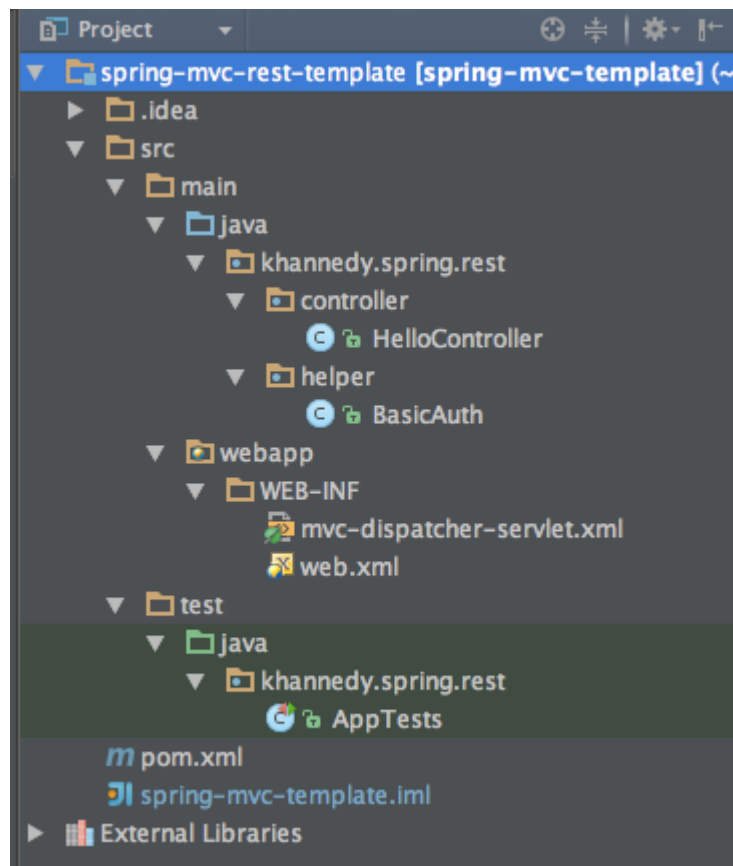
Beuh, tenang aja, semua IDE sekarang biasanya udah terintegrasi dengan Apache Maven, kayak :

- NetBeans IDE
- Eclipse IDE
- Spring Source Tool Suite
- IntelliJ IDEA

Di buku ini saya mau pake IntelliJ IDEA, tapi kalo lebih seneng pake NetBeans juga bisa kok, atau kalo mau pake Eclipse, saran gw mending pake Spring Source Tool Suite aja, udah ada plugin spring nya, kalo di Eclipse gak ada.

Untuk buka project, silahkan deh pake IDE masing2 yang ente semua pake.

Struktur project nya seperti ini kalo dibuka ti IntelliJ IDEA



Kalo mau liat gimana sih contoh hello world RESTful web service pake Spring Web MVC, bisa buka file HelloController, kayak gini isinya :

```
1 package khannedy.spring.rest.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RequestMethod;
6 import org.springframework.web.bind.annotation.ResponseBody;
7
8 @Controller
9 public class HelloController {
10
11     @ResponseBody
12     @RequestMapping(value = "/hello", method = {RequestMethod.GET})
13     public String hello() {
14         return "follow my twitter @khannedy";
15     }
16
17 }
```

Untuk membuat kelas RESTful web service pake Spring Web MVC, harus ada annotation @Controller di kelas nya, seperti digambar diatas.

Disitu ada method hello(), dimana itu gw tambahin @ResponseBody dan @RequestMapping, itu menandakan kalo metode itu merupakan salah satu

method RESTful web service. Disana @RequestMapping nya punya value = "/hello" yang artinya kalo metode itu bisa diakses via HTTP menggunakan URL

<http://blablabla.com/nama-app/hello>

blablabla.com bisa disesuaikan dengan website, dan nama-app juga disesuaikan dengan nama projectnya ya ☺

Trus ada tanda RequestMethod.GET itu artinya kalo untuk ngakses method ini, kita harus ngakses pake HTTP GET, kalo kita ubah jadi RequestMethod.POST maka harus diakses pake HTTP POST.

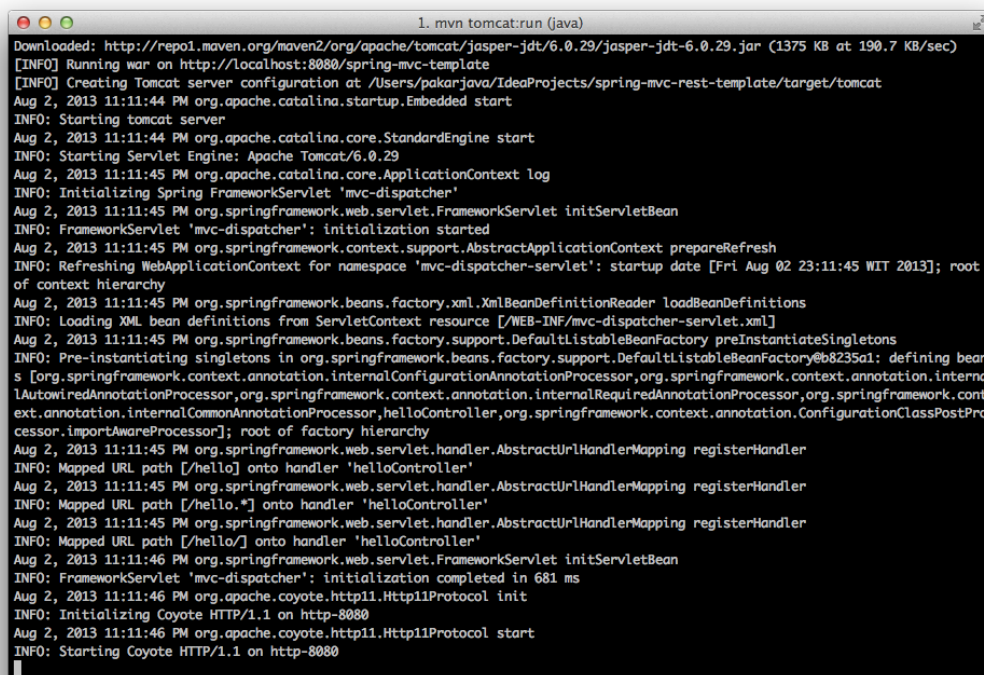
Untuk ngetestnya, kita perlu jalanin dulu projectnya, caranya?

Ngejalanin projectnya

Gw udah tambahin maven tomcat plugin di projectnya, jadi cukup pake perintah ini di terminal :

mvn clean tomcat7:run

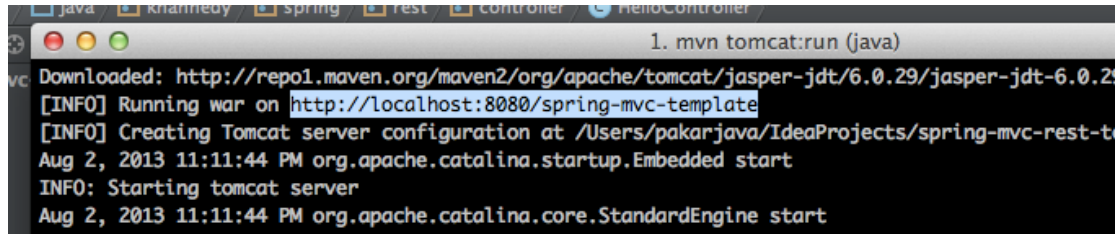
Tunggu sampai running, dan pastiin ente running perintah itu di project nya.



```
Downloaded: http://repo1.maven.org/maven2/org/apache/tomcat/jasper-jdt/6.0.29/jasper-jdt-6.0.29.jar (1375 KB at 190.7 KB/sec)
[INFO] Running war on http://localhost:8080/spring-mvc-template
[INFO] Creating Tomcat server configuration at /Users/pakarjava/IdeaProjects/spring-mvc-rest-template/target/tomcat
Aug 2, 2013 11:11:44 PM org.apache.catalina.startup.Embedded start
INFO: Starting tomcat server
Aug 2, 2013 11:11:44 PM org.apache.catalina.core.StandardEngine start
INFO: Starting Servlet Engine: Apache Tomcat/6.0.29
Aug 2, 2013 11:11:45 PM org.apache.catalina.core.ApplicationContext log
INFO: Initializing Spring FrameworkServlet 'mvc-dispatcher'
Aug 2, 2013 11:11:45 PM org.springframework.web.servlet.FrameworkServlet initServletBean
INFO: FrameworkServlet 'mvc-dispatcher': initialization started
Aug 2, 2013 11:11:45 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing WebApplicationContext for namespace 'mvc-dispatcher-servlet': startup date [Fri Aug 02 23:11:45 WIT 2013]; root
of context hierarchy
Aug 2, 2013 11:11:45 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from ServletContext resource [/WEB-INF/mvc-dispatcher-servlet.xml]
Aug 2, 2013 11:11:45 PM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@b8235a1: defining bean
s [org.springframework.context.annotation.internalConfigurationAnnotationProcessor,org.springframework.context.annotation.interna
lAutowiredAnnotationProcessor,org.springframework.context.annotation.internalRequiredAnnotationProcessor,org.springframework.cont
ext.annotation.internalCommonAnnotationProcessor,helloController,org.springframework.context.annotation.ConfigurationClassPostPro
cessor.importAwareProcessor]; root of factory hierarchy
Aug 2, 2013 11:11:45 PM org.springframework.web.servlet.handler.AbstractUrlHandlerMapping registerHandler
INFO: Mapped URL path [/hello] onto handler 'helloController'
Aug 2, 2013 11:11:45 PM org.springframework.web.servlet.handler.AbstractUrlHandlerMapping registerHandler
INFO: Mapped URL path [/hello.*] onto handler 'helloController'
Aug 2, 2013 11:11:45 PM org.springframework.web.servlet.handler.AbstractUrlHandlerMapping registerHandler
INFO: Mapped URL path [/hello/*] onto handler 'helloController'
Aug 2, 2013 11:11:45 PM org.springframework.web.servlet.handler.AbstractUrlHandlerMapping registerHandler
INFO: Mapped URL path [/hello/] onto handler 'helloController'
Aug 2, 2013 11:11:46 PM org.springframework.web.servlet.FrameworkServlet initServletBean
INFO: FrameworkServlet 'mvc-dispatcher': initialization completed in 681 ms
Aug 2, 2013 11:11:46 PM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-8080
Aug 2, 2013 11:11:46 PM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
```

Kalo udah running, silahkan buka halaman url-nya di browser, url nya apa kang?

Cek aja di log terminalnya, biasanya ada, nih kalo di terminal gw ada log seperti ini :

A terminal window titled "1. mvn tomcat:run (java)" showing the output of a Maven command. The logs indicate that the application is running on http://localhost:8080/spring-mvc-template and that the Tomcat server is starting successfully. The logs include the following text:

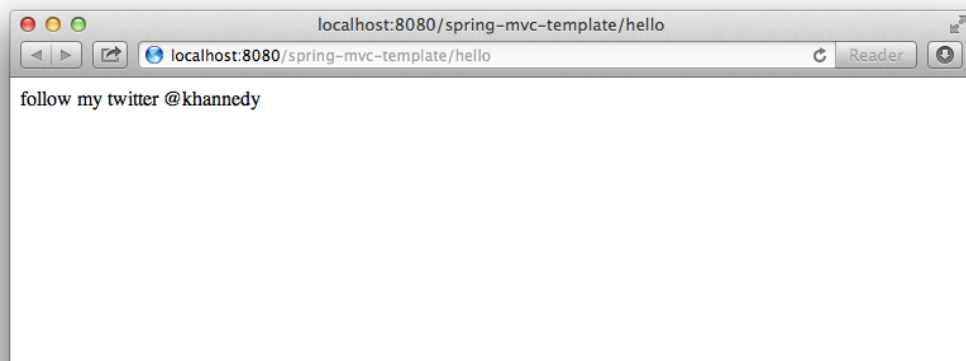
```
Downloaded: http://repo1.maven.org/maven2/org/apache/tomcat/jasper-jdt/6.0.29/jasper-jdt-6.0.29.jar
[INFO] Running war on http://localhost:8080/spring-mvc-template
[INFO] Creating Tomcat server configuration at /Users/pakarjava/IdeaProjects/spring-mvc-rest-t
Aug 2, 2013 11:11:44 PM org.apache.catalina.startup.Embedded start
INFO: Starting tomcat server
Aug 2, 2013 11:11:44 PM org.apache.catalina.core.StandardEngine start
```

<http://localhost:8080/spring-mvc-template>

Karena tadi di HelloController itu gw pake /hello, jadinya URL yang dibuka itu ini

<http://localhost:8080/spring-mvc-template/hello>

Hasilnya kayak gini nih :



Simple kan? Gampang toh, gak pake ribet ☺

Kalo gitu sekarang baru kita mulai beraksi!!!

Siap2 NGEBUL tuk OTAK!!!

Nyiapin data dulu

Sebelum kita bikin RESTful web service nya, kita siapin dulu datanya deh, gak usah pake database, tar bukannya ke panjang. Simple aja, kita bikin kelas model (data) aja.

Data apaan ya? (sumpah nih gw lagi ngetik ini blon kepikiran datanya mau apa 😊)

..... tar dah, gw makan dulu, siapa tau dapet pencerahan 😊

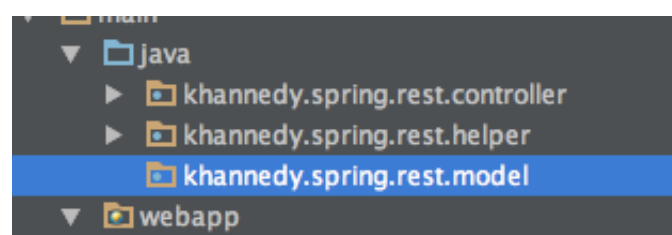
.....

Oke, datanya gak usah yang kompleks, data barang aja, kurang lebih datanya kayak gini :

Atribut	Tipe Data
Kode	String
Nama	String
Stok	Integer
Harga	Long
Mudah Terbakar	Boolean
Kategori	String
Tanggal Kadaluarsa	Date

Bikin kelas model barang-nya

Sekarang ayo kita bikin kelas Barang nya, tapi sebelumnya, silahkan bikin package model dulu, disini gw bikin package **khannedy.spring.rest.model**.



Terserah deh ente mau nama package nya apa. Gak wajib sama kok.

Kalo udah, di package itu sekarang kita bikin kelas Barang nya. Seperti ini :

```

1 package khannedy.spring.rest.model;
2
3 /**
4  * Created by pakarjava on 8/3/13.
5  */
6 public class Barang {
7
8
9
10 }
11

```

Kalo udah, tinggal lanjutin bikin semua atribut yang tadi udah gw bahas diatas.

Eng ing eng!!!! Jadi deh...

```

1 package khannedy.spring.rest.model;
2
3 import java.util.Date;
4
5 /**
6  * Created by pakarjava on 8/3/13.
7  */
8 public class Barang {
9
10     private String kode;
11     private String nama;
12     private String kategori;
13     private Long harga;
14     private Integer stok;
15     private Boolean mudahTerbakar;
16     private Date tanggalKadaluarsa;
17

```

Tinggal kita bikin **getter and setter** nya, silahkan bikin sendiri, biasanya tiap IDE punya cara cepat sendiri2, jadi jangan bikin manual CAPE COY!!

```

21
22     public String getNama() {
23         return nama;
24     }
25
26     public void setNama(String nama) {
27         this.nama = nama;
28     }
29
30     public String getKategori() {
31         return kategori;
32

```

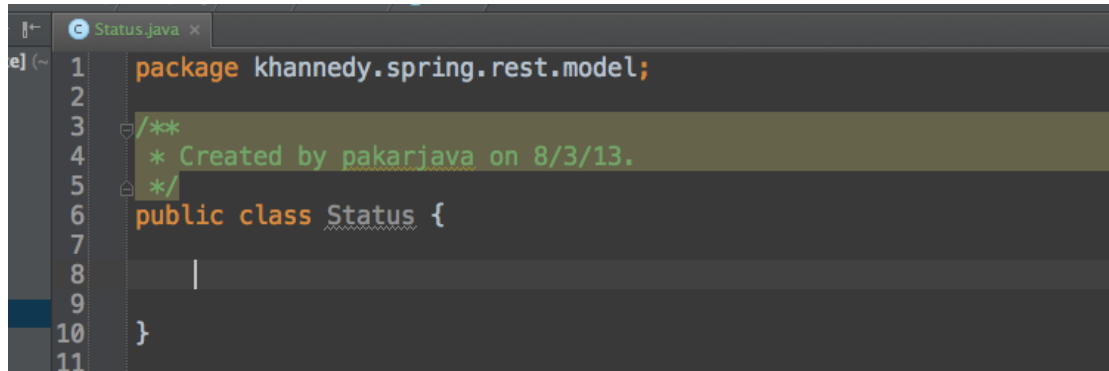
Bikin kelas status

Setelah bikin kelas model Barang, sekarang kita bikin kelas status.

Buat apa ya?

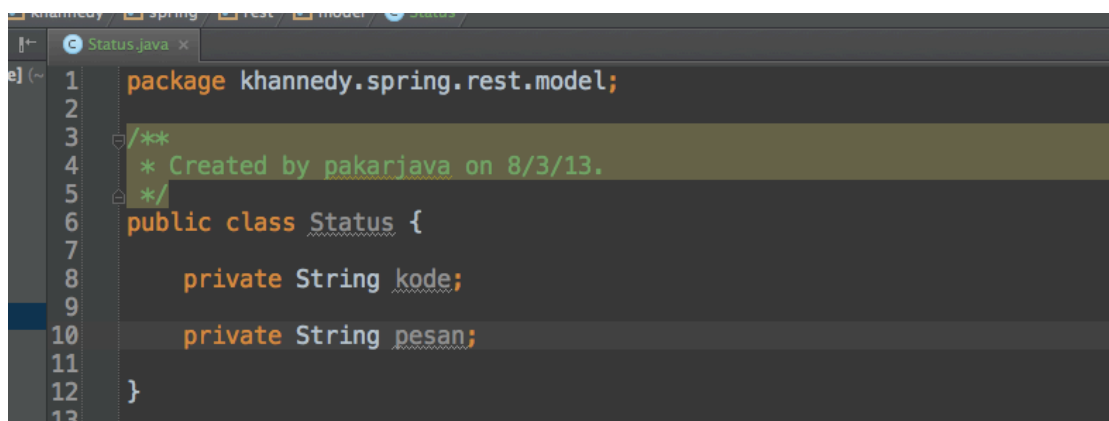
Alah jangan banyak tanya, tar juga tau sendiri, udah ikutin dulu aja, susah jelasinnya kalo soal ini.

Silahkan bikin kelas Status di package model :



```
1 package khannedy.spring.rest.model;
2
3 /**
4  * Created by pakarjava on 8/3/13.
5  */
6 public class Status {
7
8
9
10 }
11
```

Nah khusus kelas status, tambahin 2 atribut, kode dan pesan, kayak gambar dibawah ini.



```
1 package khannedy.spring.rest.model;
2
3 /**
4  * Created by pakarjava on 8/3/13.
5  */
6 public class Status {
7
8     private String kode;
9
10    private String pesan;
11
12 }
13
```

Kalo udah bikin **getter dan setter** nya lagi untuk semua atribut nya tuh.



```
12 public String getKode() {
13     return kode;
14 }
15
16 public void setKode(String kode) {
17     this.kode = kode;
18 }
19
20 public String getPesan() {
21     return pesan;
22 }
```

Bikin RESTful Web Service-nya

Sekarang saatnya bikin kelas RESTful web service nya.

Tenang tenang tenang, gak begitu menyeramkan kayak judulnya kok ☺

Bikin kelas BarangController-nya

Saya biasa bikin kelas RESTful itu dengan nama Controller, itu sih kebiasaan saya, gak usah ditiru juga gak apa2, kalo ente mau bikin kelas BarangRESTful juga gak apa2, monggo silahkan.

```
1 package khannedy.spring.rest.controller;  
2  
3 /**  
4  * Created by pakarjava on 8/3/13.  
5  */  
6 public class BarangController {  
7  
8  
9  
10 }  
11
```

Kalo udah selesai, kita tandai kalo kelas BarangController ini adalah kelas RESTful dengan nambahin annotation `@Controller` di kelas-nya, kayak gini nih :

```
1 package khannedy.spring.rest.controller;  
2  
3 import org.springframework.stereotype.Controller;  
4  
5 /**  
6  * Created by pakarjava on 8/3/13.  
7  */  
8 @Controller  
9 public class BarangController {  
10
```

Nyiapin controller-nya dulu

Supaya gak kepanjangan nih buku, jadi gw gak akan pake database, cukup kita simpen aja di memory datanya, jadi kita gunakan MAP (key-value) aja sebagai database nya.

Jadi sekarang kita tambahin dulu tuh atribut Map nya dulu, kurang lebih seperti ini nih.

```

1 package khannedy.spring.rest.controller;
2
3 import khannedy.spring.rest.model.Barang;
4 import org.springframework.stereotype.Controller;
5
6 import java.util.HashMap;
7 import java.util.Map;
8
9 /**
10  * Created by pakarjava on 8/3/13.
11  */
12 @Controller
13 public class BarangController {
14
15     private Map<String, Barang> map = new HashMap<String, Barang>();
16
17 }
18

```

Trus selain itu, perlu diketahui kalo biasanya RESTful web service itu, struktur datanya pake JSON (JavaScript Object Notation).

Gak ngerti? Belajar dulu sana...

- <http://en.wikipedia.org/wiki/JSON>
- <http://www.json.org>

Nah karena pake JSON, kita juga harus bisa konversi data JSON jadi objek JAVA, namun sayangnya itu ribetnya minta ampun ☹

Tapi tenang, dari pada kita konversi manual, lebih baik kita pake converter yang udah ada aja, disini saya pake GSON (Google JSON), library udah saya tambahkan ke project sejak awal, jadi tinggal pake aja ☺

Sekarang silahkan bikin objek Gson di kelas BarangController nya, kayak dibawah ini nih.

```

1 /**
2  *
3  */
4 @Controller
5 public class BarangController {
6
7     private Gson gson = new GsonBuilder().setDateFormat("dd/MM/yyyy").create();
8
9     private Map<String, Barang> map = new HashMap<String, Barang>();
10
11 }
12

```

Di kode diatas, gw bikin format tanggalnya jadi "dd/MM/yyyy", jadi kalo tanggal 10 Agustus 2010, jadinya datanya harus diisi dengan nilai "10/08/2010", kalo mau dirubah sih silahkan aja, gw gak maksa kok :P

Udah deh, sekarang kita FOKUS bikin method2 buat RESTful nya, dimulai dari...

Nambah data barang

Oke, sekarang yang pertama kita akan bikin method untuk nambah data barang. Simple aja, cukup kita bikin method dengan nama insert() trus ada 2 parameter, parameter pertama HttpServletRequest dan yang kedua String json.

Apa itu HttpServletRequest?

Gak ngerti? Beud dah, belajar dulu Java Web Dasar deh sana, baru balik lagi baca buku ini :P

```
21
22 public String insert(HttpServletRequest servletRequest, String json){
23     |
24 }
25
```

Habis itu, untuk nandain kalo metode yang kita bikin itu adalah RESTful method, maka kita harus nambahin beberapa annotation kayak gambar dibawah ini.

```
@ResponseBody
@RequestMapping(value = "/barang/insert", method = RequestMethod.POST)
public String insert(HttpServletRequest servletRequest, @RequestBody String json) {
}
}
```

Di method nya ditambahin @ResponseBody dan @RequestMapping. Value dari request mapping nya itu “/barang/insert” itu nanti jadi URL untuk ngakses RESTful method ini.

Biasanya untuk INSERT/CREATE (bikin data baru), biasanya memang HTTP method yang dipake adalah POST, makanya di kode diatas gw set method nya jadi RequestMethod.POST

JANGAN LUPA untuk nambahin **@RequestBody** di String json nya, buat nandain kalo nanti request JSON yang dikirim ama client itu dimasukking ke parameter itu.

Sekarang tinggal kita isi method nya sama kode-kode program untuk konversi data JSON jadi Object Barang, trus simpan ke dalam Map.

```
@ResponseBody
@RequestMapping(value = "/barang/insert", method = RequestMethod.POST)
public String insert(HttpServletRequest servletRequest, @RequestBody String json) {
    // konversi dari json ke object barang
    Barang barang = gson.fromJson(json, Barang.class);

    // simpen data barang ke map pake key kode barang
    map.put(barang.getKode(), barang);
}
}
```

Setelah itu, jangan lupa buat ngasih tau ke client saat data nya berhasil disimpan (INSERT).

Nah disinilah kita bikin JSON lagi pake object Status yang udah kita buat, misal kayak gini.

```
@ResponseBody
@RequestMapping(value = "/barang/insert", method = RequestMethod.POST)
public String insert(HttpServletRequest servletRequest, @RequestBody String json) {

    // konversi dari json ke object barang
    Barang barang = gson.fromJson(json, Barang.class);

    // simpen data barang ke map pake key kode barang
    map.put(barang.getKode(), barang);

    // bikin status sukses
    Status status = new Status();
    status.setKode("200"); // kode tersebut bisa berapa aja
    status.setPesan("Sukses nyimpen data barang");

    // bikin response json
    return gson.toJson(status);
}
```

Selesai! Sekarang kita udah implementasi RESTful method buat nambah/bikin data barang, lanjut ke method selanjutnya...

Ngambil data barang

Untuk ngambil data barang, kita sekarang bikin method dengan nama find(), dan untuk parameternya kita tambahkan HttpServletRequest dan kode barang yang akan diambil.

```
public String find(HttpServletRequest servletRequest, String kode) {

}
```

Sekarang tambahkan lagi annotation2 yang diperlukan. Eng ing eng!!!

```
@ResponseBody
@RequestMapping(value = "/barang/find/{kode}", method = RequestMethod.GET)
public String find(HttpServletRequest servletRequest,
                  @PathVariable("kode") String kode) {

    |

}
```

Sedikit berbeda dengan method insert(), kalo di method find() kita pake HTTP GET dan juga jika diperhatikan di value untuk RequestMapping, ada {kode}.

Apa sih maksudnya {kode} di URL tersebut?

Itu artinya, bahwa kode bisa berubah2, jadi misal kita ingin mengambil barang dengan kode 10106031, maka gunakan URL :

<http://blablabla.com/nama-app/barang/find/10106031>

Kalo pengen ngambil barang dengan kode K001, tinggal gunakan url

<http://blablabla.com/nama-app/barang/find/K001>

Seperti itu kurang lebih.

Oleh karena itu, di parameter kode, kita tambahkan annotation **@PathVariable("kode")**, bukan lagi **@RequestBody**.

Sekarang mari kita isi method nya.

```
@ResponseBody
@RequestMapping(value = "/barang/find/{kode}", method = RequestMethod.GET)
public String find(HttpServletRequest servletRequest,
                  @PathVariable("kode") String kode) {

    // ambil barang di map
    Barang barang = map.get(kode);

    // check apa barang ada atau enggak
    if (barang == null) {
        // kalo gak ada, return null aja
        return null;
    } else {
        // kalo ada, convert jadi JSON
        return gson.toJson(barang);
    }
}
```

Ngubah data barang

Sekarang untuk RESTful method buat ngubah data barang.

Gak perlu banyak basa basi deh ya, intinya mirip kayak insert(), kita buat aja method update() kayak gini hasilnya :


```

@ResponseBody
@RequestMapping(value = "/barang/update", method = RequestMethod.PUT)
public String update(HttpServletRequest servletRequest, @RequestBody String json) {

    // konversi dari json ke object barang
    Barang barang = gson.fromJson(json, Barang.class);

    // cek apa ada data barang dengan kode yang sama
    if (map.containsKey(barang.getKode())) {
        // kalo ada yang sama
        // update data barang dengan yang baru
        map.put(barang.getKode(), barang);

        // bikin status sukses
        Status status = new Status();
        status.setKode("200"); // kode terserah bisa berapa aja
        status.setPesan("Sukses mengubah data barang");

        // bikin response json
        return gson.toJson(status);
    } else {
        // kalo barang nya gak ada
        // bikin status gagal
        Status status = new Status();
        status.setKode("404"); // kode terserah bisa berapa aja
        status.setPesan("Barang gak ditemukan");

        // bikin response json
        return gson.toJson(status);
    }
}

```

Perlu diperhatikan, kalo ngubah data barang, biasanya di RESTful menggunakan HTTP method PUT, makanya dikode diatas gw pake RequestMethod.PUT

Ngapus data barang

Untuk ngehapus data barang, sama seperti ngambil data barang, jadi kita akan gunakan {kode}, kurang lebih kodenya seperti dibawah ini .

```

@ResponseBody
@RequestMapping(value = "/barang/delete/{kode}", method = RequestMethod.DELETE)
public String delete(HttpServletRequest servletRequest,
                    @PathVariable("kode") String kode) {

    // hapus data di map
    Barang barang = map.remove(kode);

    // check apa barang ada atau enggak
    if (barang == null) {

        // kalo gak ada, return status gagal
        Status status = new Status();
        status.setKode("404");
        status.setPesan("Barang gak ditemukan");
        return gson.toJson(status);

    } else {

        // kalo gak ada, return status sukses
        Status status = new Status();
        status.setKode("200");
        status.setPesan("Barang berhasil dihapus");
        return gson.toJson(status);

    }

}

```

Dan untuk HTTP method nya menggunakan DELETE, jadi diatas gw menggunakan RequestMethod.DELETE

Ngambil semua data barang

Method yang terakhir yang akan kita buat adalah ngambil seluruh data barang, disini kita bikin nama method nya dengan nama findAll(), dan kita gak perlu pake parameter apapun, jadi cukup seperti ini aja.

```

@ResponseBody
@RequestMapping(value = "/barang/findall", method = RequestMethod.GET)
public String findAll(HttpServletRequest servletRequest) {

    // buat data list barang
    List<Barang> list = new ArrayList<Barang>();

    // tambahin semua data barang dari map
    list.addAll(map.values());

    // return sebagai json
    return gson.toJson(list);

}

```

Selasai deh semuanya 😊

Gampang kan? Gak susah2 amet bikin RESTful web service 😊

Ngetest di client

Karena udah jadi, sekarang tinggal kita test, mudah2an gak ada error nih 😊
Soalnya gw bikinnya instan juga, kalo ada error, tar kita perbaiki lagi bareng2 😊

Sebelum ngetest, silahkan clean projectnya dulu, trus running lagi pake maven pake perintah ini :

mvn clean tomcat7:run

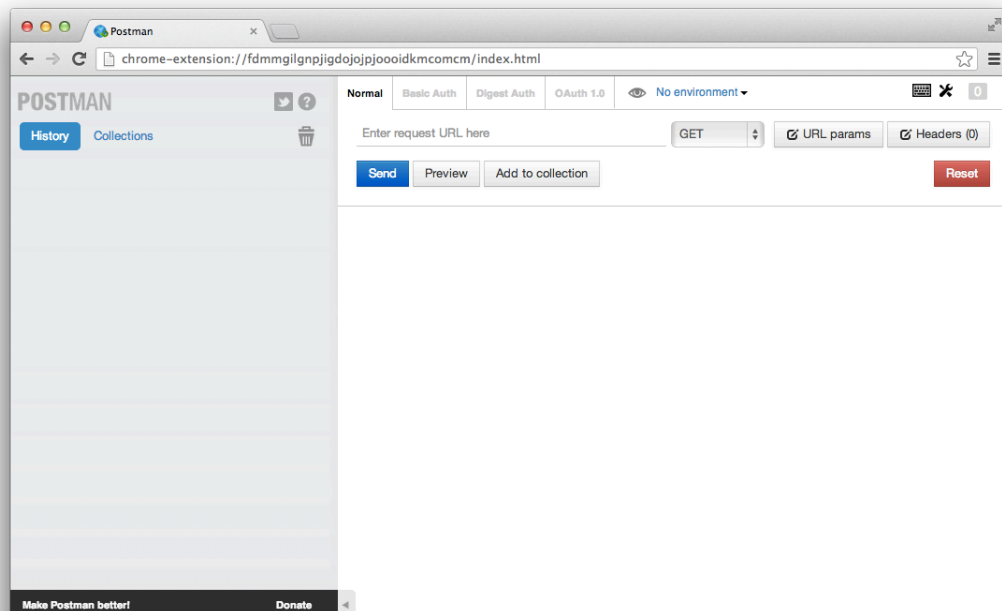
Tunggu sampai jalan!!!

Pake POSTMAN

Untuk ngetestnya, saya saranin pake REST Client, bisa pake apa aja sih sebenarnya, tapi saya saranin pake POSTMAN, soalnya itu aplikasi plugin buat Google Chrome, bisa install disini :

<https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?hl=en>

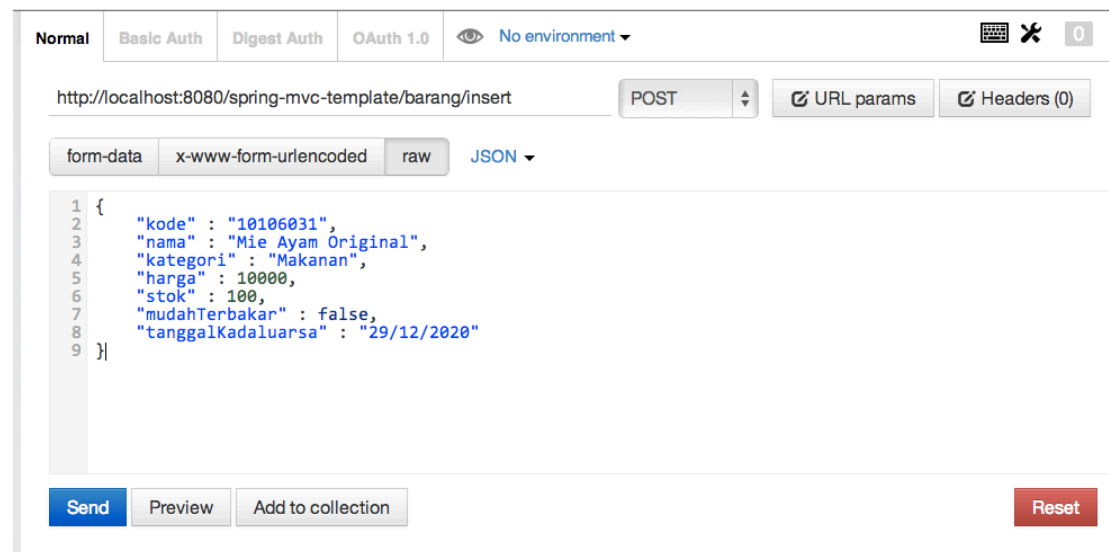
Kurang lebih aplikasinya seperti ini tampilannya



Sekarang kita coba satu persatu method2 RESTful nya, dari mulai nambah barang sampai ngambil semua barang, yu!!!

Ngetest nambah barang

Untuk ngetest nambah data barang, coba buka POSTMAN, trus isi dengan data kayak digambarkan ini deh



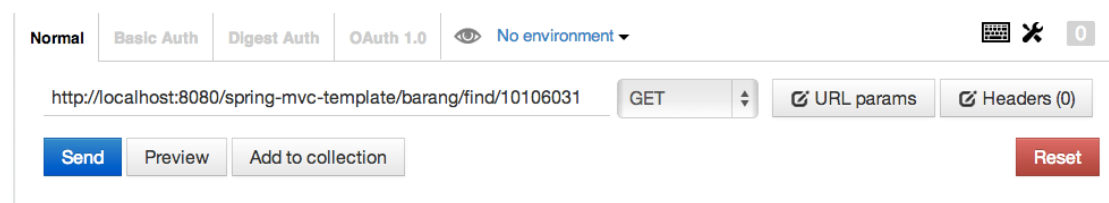
Kalo udah, klik tombol **SEND**, buat ngirim data JSON nya ke aplikasi RESTful web servicenya, nanti bakal dapat respon kayak gini :



Sekarang, tambahin deh beberapa data barang yang pengen ente tambahin, SEPUASNYA, ampe pegel jari juga gak apa2 :P

Ngetest ngambil barang

Sekarang kita coba ngambil data yang udah kita simpen sebelumnya, caranya bisa dilihat di gambar dibawah ini nih



Coba kita klik tombol Send, nanti hasilnya jadi kayak gini nih :

Normal Basic Auth Digest Auth OAuth 1.0 No environment

http://localhost:8080/spring-mvc-template/barang/find/10106031 GET URL params Headers (0)

Send Preview Add to collection Reset

Body Headers (4) STATUS 200 OK TIME 40 ms

Pretty Raw Preview JSON XML

```
1 {
2   "kode": "10106031",
3   "nama": "Mie Ayam Original",
4   "kategori": "Makanan",
5   "harga": 10000,
6   "stok": 100,
7   "mudahTerbakar": false,
8   "tanggalKadaluarsa": "29/12/2020"
9 }
```

Weiii, gampang kan? ☺

Ngetest ngubah barang

Sekarang, kita coba ngetest ngubah data barang yang udah di tambah, misal kita ubah data barang dengan kode **10106031**.

Normal Basic Auth Digest Auth OAuth 1.0 No environment

http://localhost:8080/spring-mvc-template/barang/update PUT URL params Headers (0)

form-data x-www-form-urlencoded raw JSON

```
1 {
2   "kode" : "10106031",
3   "nama" : "Sirup CBA",
4   "kategori" : "Minuman",
5   "harga" : 15000,
6   "stok" : 1000,
7   "mudahTerbakar" : false,
8   "tanggalKadaluarsa" : "29/12/2020"
9 }
```

Send Preview Add to collection Reset

Kalo kita SEND, hasilnya pastiin sukses, kalo gagal berarti kode yang kita kirim salah tuh.

Body Headers (4) STATUS 200 OK TIME 53 ms

Pretty Raw Preview JSON XML

```
1 {
2   "kode": "200",
3   "pesan": "Sukses mengubah data barang"
4 }
```

Buat mastiin kalo memang bener data barang berubah, kita cek aja pake find() lagi, nih hasilnya

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/spring-mvc-template/barang/find/10106031
- Status:** 200 OK
- Time:** 43 ms
- Body:** A JSON object representing a product:

```
1 {
2   "kode": "10106031",
3   "nama": "Sirup CBA",
4   "kategori": "Minuman",
5   "harga": 15000,
6   "stok": 1000,
7   "mudahTerbakar": false,
8   "tanggalKadaluarsa": "29/12/2020"
9 }
```

Kayaknya berhasil ☺ hehehehe...

Ngetest ngapus barang

Sekarang coba kita hapus salah satu datanya, caranya coba kayak gambar dibawah ini nih

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:8080/spring-mvc-template/barang/delete/10106031
- Body:** A single line of text: 1

Klik SEND, pastiin ngehapus nya sukses

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:8080/spring-mvc-template/barang/delete/10106031
- Status:** 200 OK
- Time:** 49 ms
- Body:** A JSON object representing a success message:

```
1 {
2   "kode": "200",
3   "pesan": "Barang berhasil dihapus"
4 }
```

Buat mastiin kalo datanya udah ilang, kita coba find() lagi deh.

Normal Basic Auth Digest Auth OAuth 1.0 No environment

http://localhost:8080/spring-mvc-template/barang/find/10106031 GET URL params Headers (0)

Send Preview Add to collection Reset

Body Headers (3) STATUS 200 OK TIME 30 ms

Pretty Raw Preview JSON XML

1

Dan hasilnya ternyata KOSONG!!! Yeah berhasil!!! 😊

Ngetest ngambil semua data barang

Sekarang kita coba liat seluruh data barang yang udah ada nih, caranya kayak gambar dibawah ini nih

Normal Basic Auth Digest Auth OAuth 1.0 No environment

http://localhost:8080/spring-mvc-template/barang/findall GET URL params Headers (0)

Send Preview Add to collection Reset

Body Headers (4) STATUS 200 OK TIME 39 ms

Pretty Raw Preview JSON XML

Klik SEND, pastiin semua datanya muncul

Body Headers (4) STATUS 200 OK TIME 39 ms

Pretty Raw Preview JSON XML

```

1 [
2   {
3     "kode": "10106035",
4     "nama": "Sirup CBA",
5     "kategori": "Minuman",
6     "harga": 15000,
7     "stok": 1000,
8     "mudahTerbakar": false,
9     "tanggalKadaluarsa": "29/12/2020"
10  },
11  {
12    "kode": "10106033",
13    "nama": "Sirup CBA",
14    "kategori": "Minuman",
15    "harga": 15000,
16    "stok": 1000,
17    "mudahTerbakar": false,
18    "tanggalKadaluarsa": "29/12/2020"
19  },
20  {
21    "kode": "10106034",
22    "nama": "Sirup CBA",
23    "kategori": "Minuman",
24    "harga": 15000,
25    "stok": 1000.

```

Yeah, berhasil, berarti sekarang kita udah berhasil bikin RESTful web service, MANTAP CUY!

Selesai!

Mantap dah, udah selesai juga nih buku 😊, tapi perlu diketahui kalo bikin aplikasi, sebenarnya gak selesai sampai disini lho

Lho? Trus apa yang belum selesai?

Apa nih yang belum selesai?

Banyak, kayak autentikasi nya gimana? Kalo bulet2 bikin aplikasi RESTful web service nya gini, ya gampang di hek sama orang, wong gak ada autentikasi username password kok.

Trus gimana?

Tenang kita bikin sekarang.

Perlu diketahui, kalo biasanya autentikasi di RESTful itu standard nya pake BASIC AUTH, ada juga pake OAuth dan lain-lain, tapi saya rekomendasi pake BASIC AUTH soalnya simple, jadi saya juga gak panjang2 bikin bukunya 😊 #hehe

Apaan tuh BASIC AUTH? Buset deh, belajar dulu sana!

http://en.wikipedia.org/wiki/Basic_access_authentication

Nambahin autentikasi pake BASIC AUTH

Untuknya udah gw sedian sebuah kelas dengan nama BasicAuth buat ngedeteksi data BASIC AUTH, jadi ente gak harus bikin manual.

Sekarang coba bikin sebuah method di BarangController buat ngecek data username dan password nya, kurang lebih kayak gini nih

```
public boolean checkAccess(HttpServletRequest servletRequest) {  
    // misal username dan password harus ...  
    String username = "eko";  
    String password = "@khannedy";  
  
    // buat object BasicAuth  
    BasicAuth basicAuth = new BasicAuth(servletRequest);  
  
    // check username dan password  
    return username.equals(basicAuth.getUsername()) &&  
           password.equals(basicAuth.getPassword());  
}
```


Sekarang di setiap method nya, dari mulai insert(), update(), delete(), find() dan findAll(), tambahkan kode kayak gini nih buat ngecek username dan password.

```
// check hak akses
if (!checkAccess(servletRequest)) {
    Status status = new Status();
    status.setKode("408");
    status.setPesan("Hak akses ditolak");
    return gson.toJson(status);
}
```

Inget ya, di semua method, contoh kalo di insert() jadi gini nih :

```
@ResponseBody
@RequestMapping(value = "/barang/insert", method = RequestMethod.POST)
public String insert(HttpServletRequest servletRequest, @RequestBody String json) {

    // check hak akses
    if (!checkAccess(servletRequest)) {
        Status status = new Status();
        status.setKode("408");
        status.setPesan("Hak akses ditolak");
        return gson.toJson(status);
    }

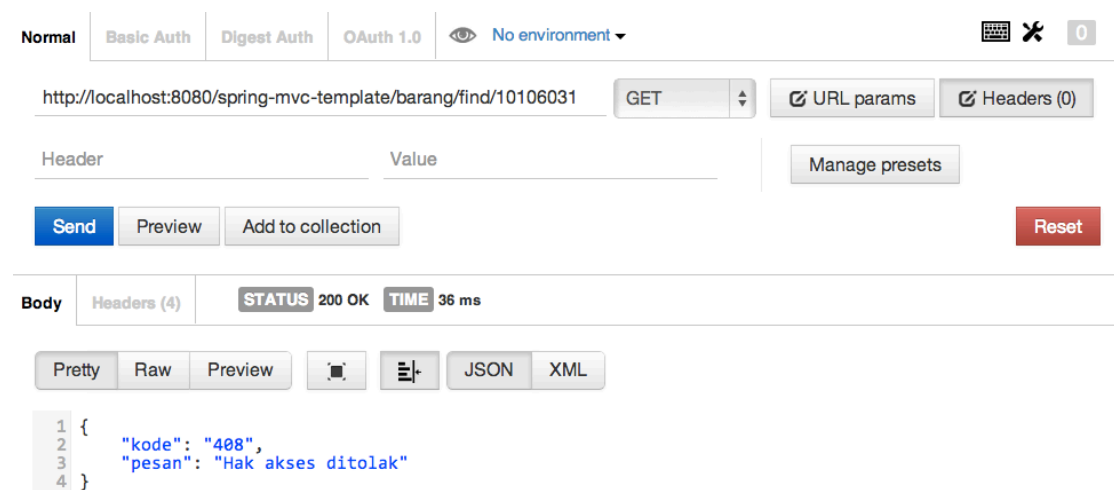
    // konversi dari json ke object barang
    Barang barang = gson.fromJson(json, Barang.class);

    // simpen data barang ke map pake key kode barang
    map.put(barang.getKode(), barang);
}
```

Sekali lagi gw bilang, HARUS DI SEMUA METHOD!!!!

Ngetest lagi yang pake BASIC AUTH

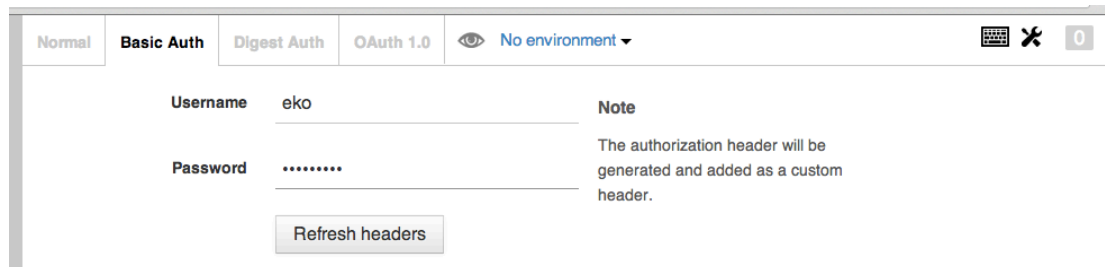
Nah sekarang kita coba ngetest lagi pake POSTMAN, misal ngambil data barang, kayak gambar dibawah ini nih



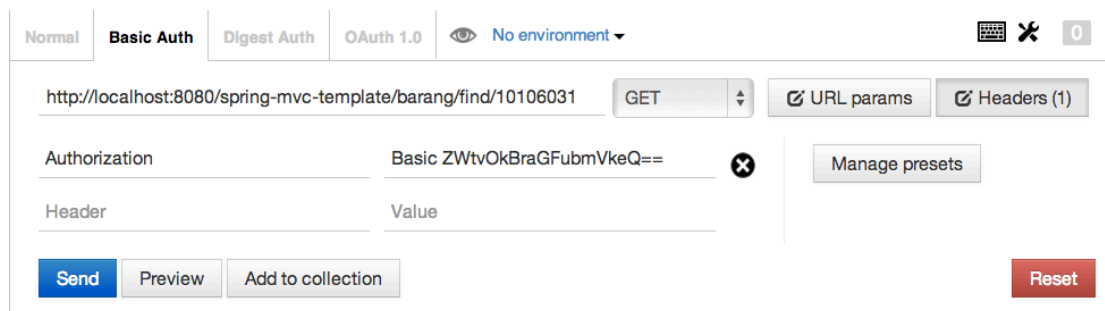
Hasilnya adalah “Hak akses ditolak”, kenapa?

Ya karena kita gak masukin username dan password. Jadi mulai sekarang kita harus masukin username dan password, gimana caranya? Ya pake BASIC AUTH donk.

Kalo di POSTMAN gampang, tinggal masuk ke tab Basic Auth, trus masukin Username dan Password nya :



Klik REFRESH HEADERS, nanti otomatis POSTMAN akan masukin tuh username dan password ke HEADER HTTP REQUEST, hasilnya kayak gini nih



Jadi ada header Authorization 😊

Sekarang klik tombol SEND, hasilnya..... ENG ING ENG!!!



HORE!!! BERHASIL!!!! 😊

Sekarang RESTful Web Service yang kita bikin udah AMAN NYAMAN DAN TERKENDALI #alah 😊

Selesai nih?

Yup, udah selesai, tapi ada beberapa yang belum dan GAK AKAN dibahas dibuku ini, contohnya :

- Gimana cara ngakses RESTful Web Service ini di aplikasi Desktop
- Atau di aplikasi Mobile kayak Android misalnya

Nah pertanyaan2 itu, akan kita bahas dibuku2 saya selanjutnya 😊 #hehehe

Keep UP 2 DATE dengan buku2 terbaru saya di twitter saya @khannedy 😊

Download kodenya dimana?

Hehe, sengaja saya kasih link download kodenya di bagian paling akhir, supaya dari awal coding dulu sendiri, kalo udah mentok, baru deh download kodenya 😊

Yang mau download codenya silahkan download disini :

<https://github.com/khannedy/spring-rest-final/archive/master.zip>

Buku siapa nih?

Ini buku milik yang baca lah ☺

Tapi yang nulis itu namanya Eko Khannedy, JURAGAN MIE AYAM, CODER JAVA, dan juga sekarang lagi jadi SECRET AGENT di salah satu PAYMENT GATEWAY di INDONESIA ☺

Saya rutin nulis artikel di <http://eecchhoo.wordpress.com> dan juga nulis buku. Setiap buku baru yang saya tulis biasanya saya publish di blog saya itu.



Untuk tips dan trik seputar CODING JAVA, atau juga sedikit tentang JURAGANPRENEUR ☺ bisa follow twitter saya di <http://twitter.com/khannedy> , selain itu di twitter juga saya sering bagi2 buku java gratis dengan tag #BukuJavaGratis

Yang mau #BukuJavaGratis lainnya, silahkan follow twitter @khannedy, trus mention saya, nanti saya DM link2 buat download #BukuJavaGratis lainnya ☺

Semoga berguna ya

Salam JURAGAN JAVA ☺