UMSIDA

PENGEMBANGAN LAYANAN WEB SERVICE DENGAN MICROFRAMEWORK

IRWAN A. KAUTSAR, Ph.D

Program Studi Informatika
Fakultas Sains & Teknologi

Universitas Muhammadiyah Sidoarjo

BUKU AJAR
Pengembangan Layanan Web Service dengan Microframework

Oleh
Irwan Alnarus Kautsar, S.Kom., M.Kom., Ph.D

Diterbitkan oleh

UMSIDA PRESS

BUKU AJAR
Pengembangan Layanan Web Service dengan Microframework

Penulis :

Irwan Alnarus Kautsar, Ph.D

ISBN : 978-623-7578-53-6

Editor:
Septi Budi Sartika
M. Tanzil Multazam
Copy Editor :
Fika Megawati
Design Sampul dan Tata Letak :
Mochamad Nashrullah

Penerbit :
UMSIDA Press

Redaksi :
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No 666B
Sidoarjo, Jawa TImur

Cetakan pertama (elektronik) Oktober 2019

Kata Pengantar

Bismillahirrohmanirrohim

Segala puji syukur kami panjatkan kepada Allah Azza wa Jalla atas segala rahmat-Nya dan kemudahan-Nya untuk dapat menyelesaikan buku ajar Pengembangan Layanan Web Service dengan Microframework. Ucapan terima kasih penulis berikan kepada berbagai pihak yang telah membantu dan berkontribusi baik moril maupun pemikiran. Buku ajar Layanan Web Service dengan Microframework, disusun dalam rangka menunjang mata kuliah Pengembangan Aplikasi Berbasis Web yang diharapkan menambah skill mahasiswa dalam era Industri 4.0, khususnya mahasiswa Program Studi Informatika, Fakultas Teknik Universitas Muhammadiyah Sidoarjo. Diharapkan adanya buku ajar ini, mahasiswa Program Studi Informatika mampu merancang bangun teknologi berbasis web untuk menyelesaikan masalah yang ada di lingkungan sekitar.

Penulis menyadari bahwa dalam penyusunan buku ajar ini terdapat kekurangan dan jauh dari kata sempurna. Oleh sebab itu, penulis berharap kritik, saran serta usulan demi perbaikan dan kesempurnaan buku ajar berikutnya. Mengingat tidak ada sesuatu yang sempurna tanpa saran yang membangun.

Sidoarjo, 18 Agustus 2019

Irwan A. Kautsar, S.Kom., M.Kom., Ph.D

surel: irwan@umsida.ac.id

Daftar Isi

Daftar Table

Daftar Gambar

Capaian Pembelajaran Mata Kuliah

Bab 1. Teknologi Web

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu mendeskripsikan perbedaan aplikasi Desktop dan Mobile serta Teknologi Web
- Mahasiswa mampu mendeskripsikan ciri khas dari teknologi Web.
- Mahasiswa mampu mendefinisikan teknologi-teknologi yang menunjang layanan teknologi web untuk pengolahan data hingga menjadi sebuah informasi
- Mahasiswa mampu menjelaskan arah pemanfaatan teknologi Web

Bab 2. Layanan Web (Web Service)

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu mendeskripsikan peran Web Service dalam rangka pengembangan aplikasi web
- Mahasiswa mampu mendefinisikan SOA dan SOAP
- Mahasiswa mampu menjelaskan pemanfaatan file XML dan JSON dalam layanan Web.
- Mahasiswa mampu mengimplementasikan penggalian data pada sumber Web Service

Bab 3. Framework Pengembangan Aplikasi Web

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu menerangkan peran Framework dalam rangka pengembangan aplikasi web
- Mahasiswa mampu mengklasifikasikan Komponen Front-End Layer dan Back-End Layer
- Mahasiswa mampu menjelaskan peran pengembang Front-End Developer dan Back-End.
- Mahasiswa mampu mengimplementasikan Gaya Pengembangan Model View Controller.

Bab 4.Microframework

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu menerangkan peran Micro Framework dalam pengembangan teknologi web.
- Mahasiswa mampu menjelaskan fungsi routing pada teknologi Web yang ada pada saat ini.
- Mahasiswa mampu memanfaatkan mesin template untuk renderisasi file HTML.
- Mahasiswa mampu megembangkan aksi Create Read Update dan Delete pada halaman Hypertext Markup Language (HTML) dan mesin Database.

Bab 5. Keamanan Layanan Web/Web Service Security

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu menunjukkan celah-celah (holes) pada teknologi web.
- Mahasiswa mampu menunjukkan celah keamanan pada layanan web.
- Mahasiswa mampu menjelaskan fungsi Tokenisasi pada layanan Web.
- Mahasiswa mampu memanfaatkan modul/API dalam pengamanan layanan Web.

Bab 6. Implementasi Web Service: Teknologi Web API

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu menjelaskan peran API pada layanan Web
- Mahasiswa mampu melakukan operasi data menggunakan API melalui protokol HTTP dan halaman HTML.

Bab 1. Teknologi Web

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu mendeskripsikan perbedaan aplikasi Desktop dan Mobile serta Teknologi Web
- Mahasiswa mampu mendeskripsikan ciri khas dari teknologi Web.
- Mahasiswa mampu mendefinisikan teknologi-teknologi yang menunjang layanan teknologi web untuk pengolahan data hingga menjadi sebuah informasi
- Mahasiswa mampu menjelaskan arah pemanfaatan teknologi Web

Kunci dari teknologi informasi ialah bagaimana teknologi tersebut dapat menggali, mengolah dan mengelola data menjadi sebuah informasi [1]. Contoh: terdapat suatu peluang untuk menawarkan jasa antar jemput dan antar barang menggunakan kendaraan roda dua. Untuk dapat memberikan layanan jasa tersebut, pengembang aplikasi (dalam hal ini kita sebut sebagai inventor), perlu untuk menganalisa dan memikirkan bagaimana menggali data orang yang membutuhkan jasa tersebut. Kemudian memikirkan bagaimana mengolah data tersebut. Oleh sebab itu, lulusan Program Studi Informatika diharapkan dapat memiliki kemampuan untuk membuat suatu inovasi terkait teknologi informasi tersebut.

Teknologi Informasi yang dikembangkan, sebagian besar dalam bentuk sebuah software/ aplikasi. Untuk aplikasi itu sendiri, terdapat beberapa macam bentuk apabila dilihat dari bagaimana pengguna akan mengakses informasi tersebut atau bagaimana pengguna akan menggunakan aplikasi tersebut. Apabila pengguna akan mengakses informasi menggunakan perangkat komputer seperti Laptop/PC, maka aplikasi yang dikembangkan dapat berupa aplikasi berbasis Desktop dan Aplikasi Berbasis Teknologi Web. Namun apabila pengguna akan mengakses informasi melalui perangkat mobile, maka pengembang perlu membuat aplikasi tersebut sesuai dengan platform perangkat mobile yang akan digunakan

oleh pengguna tersebut. Untuk itu inventor/ developer/pengembang perlu melakukan perencanaan dalam mengembangkan aplikasi dengan memperhatikan bagaimana aplikasi tersebut akan diimplementasikan. Hal ini disebabkan karena perbedaan platform aplikasi berbeda pula teknologi atau alat bantu pengembangan aplikasinya [2]. Sebagai contoh, untuk mengembangkan aplikasi untuk pada perangkat mobile pada sistem operasi IOS menggunakan bahasa pemrograman Swift atau Objective C. Sedangkan untuk implementasi aplikasi mobile pada sistem operasi Android menggunakan bahasa pemrograman Java atau Kotlin. Untuk itu perlu dikenalkan ciri khas/

perbedaan aplikasi Desktop dan Mobile serta Teknologi Web.

1.1. Aplikasi Desktop dan Mobile

Ciri dari aplikasi Desktop ialah aplikasi tersebut harus diinstall pada komputer pengguna. Artinya, pengembang perlu "membungkus" (bahasa teknisnya meng-compile) aplikasi tersebut agar dapat di install pada perangkat komputer yang akan digunakan pada pengguna. Yang perlu diperhatikan adalah sistem operasi yang terpasang pada perangkat komputer pengguna tersebut. Hal ini berpengaruh pada bahasa pemrograman yang digunakan untuk mengembangkan aplikasi tersebut. Beberapa bahasa pemrograman tidak bersifat multiplatform. Sebagai contoh,

Microsoft Visual Basic .NET yang menggunakan bahasa pemrograman Basic. Program yang dikembangkan dari platform tersebut setelah di compile akan menjadi sebuah file berekstensi .exe atau .msi. File .exe dan .msi hanya dapat pada sistem operasi windows.

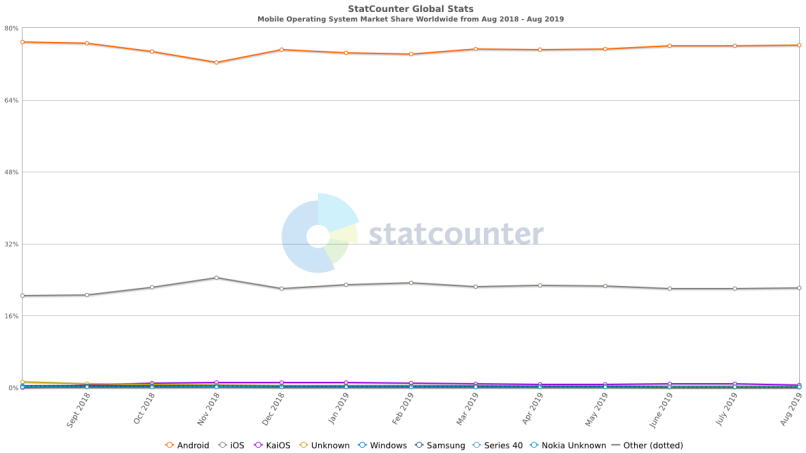
Karena setiap aplikasi yang dikembangkan dan harus diinstall pada perangkat pengguna dapat disebut aplikasi desktop. Maka dalam implementasi di lapangan, apabila pengembang mengembangkan aplikasi karena ada order dari pengguna, maka pengembang perlu menyesuaikan dengan perangkat yang dimiliki oleh pengguna. Namun, sebaliknya, apabila pengguna ingin menggunakan aplikasi yang telah disediakan pengembang atau aplikasi tersebut

merupakan mass product (seperti Microsoft Office, Aplikasi Peer to Peer, PDF Reader) maka pengguna lah yang harus menyesuaikan aplikasi yang akan diinstall. Keterbatasan pada sistem operasi inilah yang menjadikan aplikasi berbasis web lebih banyak dikembangkan daripada aplikasi Desktop. Akan tetapi, kelebihan dari aplikasi Desktop adalah melindungi kode sumber (source code) dari pengembang. Ya, setelah aplikasi Desktop "dibungkus" menjadi file .exe atau .msi, maka ketika aplikasi tersebut tidak dapat dilihat kode sumbernya. Dari aspek ekonomi, aplikasi Dekstop lebih dapat melindungi resep rahasia dari keunggulan suatu produk. Namun, dari aspek sosial, sumber tertutup tidak memberikan "edukasi" pada

pengguna. Karena kode sumber yang tertutup, tidak dapat dievaluasi oleh komunitas/pakar teknologi terkait kode tersebut. Dimungkinkan, aplikasi pada sumber tertutup secara langsung memiliki hak akses untuk mengakses data-data privat pada perangkat komputer pengguna. Dari sinilah perdebatan “close source” dan “open source” dimulai.

Aplikasi Mobile, penggunaannya mirip seperti aplikasi Desktop. Dimana pengguna harus mengunduh dan menginstall dahulu aplikasi yang diinginkan, untuk dapat digunakan pada perangkat mobile-nya. Untuk saat ini, sistem operasi pada perangkat mobile di dominasi oleh Sistem Operasi Android dan IOS. Gambar 1.1. menunjukkan dalam waktu satu

tahun terakhir, Sistem Operasi Android menguasai lebih dari 60% perangkat mobile yang ada (sumber: statcounter.com).



Gambar 1.1. Mobile OS Market Share

Dengan melihat pasar tersebut, maka kebutuhan pengembang/developer Sistem Android dan IOS masih sangat dibutuhkan. Perbedaan Aplikasi Desktop dan Aplikasi

Mobile, selain dari bahasa yang digunakan untuk mengembangkan aplikasinya, pembaca perlu memahami bahwa untuk aplikasi mobile, akses ke database tidak seperti aplikasi desktop. Yaitu apabila pada aplikasi Desktop, akses ke database dapat langsung melalui "connector" yang disediakan mesin database. Untuk aplikasi mobile, akses database hanya bisa melalui sebuah URL/Protokol HTTP (Hypertext Transfer Protocol).

Untuk itu, dalam pengembangan aplikasi Desktop dan Mobile, pengembang perlu "belajar" bahasa pemrograman yang mendukung pengembangan aplikasi Desktop dan Mobile tersebut. Dan aplikasi yang telah dikembangkan perlu dilakukan proses "instalasi". Dari satu sisi

kelebihan aplikasi Desktop dan Mobile adalah melindungi kode sumber pengembang aplikasi, namun pengembang dan pengguna perlu memperhatikan antara jenis aplikasi yang akan diinstall serta sistem operasi yang akan digunakan.

Hal ini berbeda dengan aplikasi yang berbasis Teknologi Web. Teknologi Web memungkinkan pengguna untuk menggunakan aplikasi langsung tanpa menginstall aplikasi. Untuk penjelasan lebih detail mengenai aplikasi web, dijelaskan pada sub bab. 1.2.

1.2. Aplikasi Berbasis Web

Teknologi web dikenalkan pertama kali oleh SIr Tim BErner Lee di tahun. Dimana teknologi web pertama kali dikenalkan dengan memanfaatkan jaringan teknologi internet yang telah dikenal beberapa tahun sebelumnya [3,5,9]. Konsep awal teknologi web ialah setiap komputer yang terhubung jaringan internet dapat berbagi/sharing file menggunakan protokol tertentu. Sehingga pertukaran informasi antara satu komputer dengan komputer yang lain dapat dilakukan secara cepat dan real time. Untuk mengembangkan teknologi web, pengembang tidak perlu memperhatikan platform calon pengguna. Karena untuk menggunakan aplikasi berbasis web, pengguna

cukup menggunakan aplikasi yang umumnya dikenal dengan web browser dan mengakses laman web dimana aplikasi tersebut tersimpan. Bahasa pemrograman aplikasi berbasis web terdiri dari beberapa macam. Jika dilihat dari jenisnya terdapat 2 macam. Yaitu client side scripting dan server side scripting. Client side scripting adalah bahasa pemrograman yang diproses oleh komputer klien. Sedangkan bahasa server side scripting diproses oleh komputer server dimana file-file aplikasi web tersimpan. Contoh yang termasuk dalam client side scripting adalah bahasa HTML dan Javascript. Untuk server side scripting contohnya PHP, Python dan ASP. Contoh Client side scripting/file HTML ditunjukkan pada Script #1-

001. Untuk contoh Server side scripting/file PHP ditunjukkan pada Script #1-002.

Script #1-001.

```
<!DOCTYPE html>
<html>
<body>

<h1>Hello HTML page</h1>

<p> Hello From HTML files. </p>

</body>
</html>
```

Script #1-002.

```
<?php
echo "Hello From PHP files.";

$color = "Blue";
echo " And Sky is ".
$color.".";

?>
```

Perbedaan mendasar dari Client Side Scripting dan Server side scripting selain mesin siapakah yang memproses script tersebut adalah apakah kode tersebut “terbaca” oleh client atau tidak. Untuk lebih memahami perbedaan dari kedua jenis tersebut, pembaca disarankan melakukan percobaan pada Kegiatan 1.1 dan Kegiatan 1.2.

Saat ini banyak tersedia framework untuk pengembangan aplikasi web yang berasal dari bahasa pemrograman untuk server side scripting dan client side scripting. Beberapa Framework yang digunakan untuk mengembangkan aplikasi

web baik termasuk server/client side scripting ditunjukkan pada Tabel 1.1.

Tabel 1.1. Web Framework Development

No	Framework	Sintaks	Tipe Script
1	Qcodo	PHP	Server Side
2	Phalcon	PHP	Server Side
3	CodeIgniter	PHP	Server Side
4	Yii PHP Framework	PHP	Server Side
5	Pyramid	Python	Server Side
6	Django	Python	Server Side
7	Flask	Python	Server Side
8	Ember.Js	JavaScript	Client Side
9	Backbone.js	JavaScript	Client Side
10	Meteor.Js	JavaScript	Client Side

1.3. GET dan POST

Dalam teknologi web itu sendiri, terdapat dua mekanisme dasar dalam hal request (baca: permintaan/pengiriman data) dari Client kepada Server. Yaitu GET dan POST.

GET - Client meminta data dari Server

POST - Client mengirimkan data untuk diproses oleh Server

Beberapa hal yang harus diperhatikan dalam menggunakan metode GET yaitu:

1. Sintaks pada metode GET tersimpan di Browser. Sehingga bisa di-bookmark dan di-cache.
2. Terdapat batasan panjang sintaks pada metode GET

3. Tidak direkomendasikan menggunakan metode GET untuk penggunaan data sensitif.
4. Direkomendasikan hanya digunakan untuk mengambil data dari Server

Sedangkan pada metode POST adalah:

1. Sintaks pada metode POST tidak dapat di-”bookmark”.
2. Tidak terdapat batasan pada panjang sintaks metode POST.
3. Sintaks metode POST tidak pernah di-cache oleh browser dan tidak tersimpan pada history browser.
4. Direkomendasikan untuk mengirimkan data menggunakan metode POST.

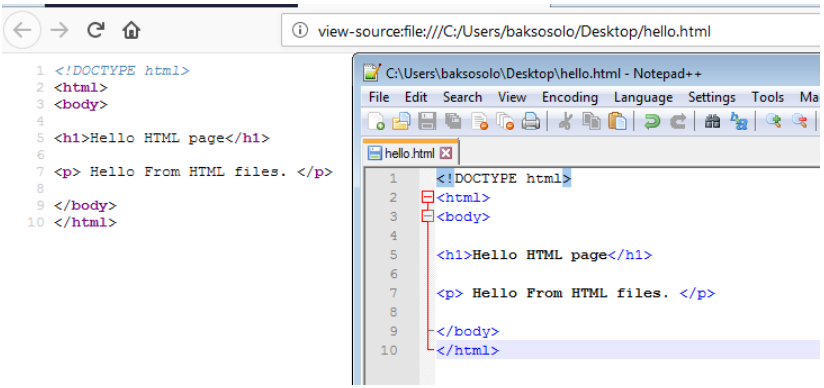
Kegiatan 1.1
Uji Coba Client Side Scripting

Alat dan Bahan:
Text Editor, Web Browser (Mozilla Firefox)

Petunjuk Uji Coba:

1. Buka Text Editor yang tersedia.
2. Simpan script #001 dengan nama hello.html di Desktop.
3. Buka hello.html yang ada di Desktop dengan web browser.
4. Klik kanan pada layar di web browser. View Page Source.
5. Amati kode yang ditampilkan. Dimana semua script ditampilkan pada web browser sama persis dengan yang ditulis pada hello.html

Ouput:

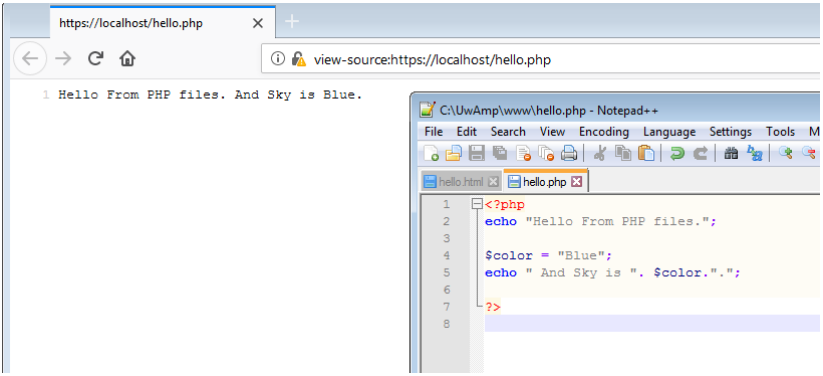

view-source:file:///C:/Users/baksosolo/Desktop/hello.html
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Hello HTML page</h1>
6
7 Hello From HTML files.
8
9 </body>
10 </html>
C:\Users\baksosolo\Desktop\hello.html - Notepad++
File Edit Search View Encoding Language Settings Tools Ma
hello.html
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Hello HTML page</h1>
6
7 Hello From HTML files.
8
9 </body>
10 </html>

Kegiatan 1.2
Uji Coba Server Side Scripting

Alat dan Bahan:
Text Editor, Web Browser, Web Server (Apache)/ WAMP Server (https://www.uwamp.com/en/)

Petunjuk Uji Coba:
1. Buka Text Editor yang tersedia.
2. Simpan script #002 dengan nama hello.php pada web server direktori. Untuk windows simpan di htdocs/Folder WWW web server. Linux /var/www.
3. Aktifkan Web Server.
4. Buka hello.php dengan web browser dengan alamat :https://localhost/hello.php
5. Klik kanan pada layar. View Page Source
6. Amati kode yang ditampilkan. Dimana kode di dalam tag <?php ?> tidak ditampilkan.

Output:


https://localhost/hello.php
view-source:https://localhost/hello.php
1 Hello From PHP files. And Sky is Blue.
C:\UwAmp\www\hello.php - Notepad++
File Edit Search View Encoding Language Settings Tools M
hello.html
hello.php
1 <?php
2 echo "Hello From PHP files.";
3
4 $color = "Blue";
5 echo " And Sky is ". $color.".";
6
7 ?>
8

Pada dasarnya, Client Side Scripting dan Server Side Scripting, kodenya bersifat open source. Atyinya dapat dilihat kode sumbernya. Namun perbedaannya, client side scripting kodenya dapat dibuka/dibaca/dipelajari pada komputer client (komputer yang mengakses file tersebut dan memproses client side scriptingnya). Sedangkan Server side scripting kodenya hanya bisa dibaca dari mesin server. Karena server side scripting diproses oleh mesin servernya. Bukan mesin client. Silahkan mencoba kegiatan 1.3.

Script #1-003

```
<!DOCTYPE html>
<html>
<body>

<h1>Hello HTML page</h1>

<p> Hello From HTML files. </p>

<?php
echo "Hello From PHP files.";

$color = "Blue";
echo " And Sky is ". $color.".";

?>

</body>
</html>
```

Kegiatan 1.3

Uji Coba Client Side Scripting dan Server Side Scripting

Alat dan Bahan:

Text Editor, Web Browser, Web Server (Apache)/ WAMP Server (https://www.uwamp.com/en/)

Petunjuk Uji Coba:

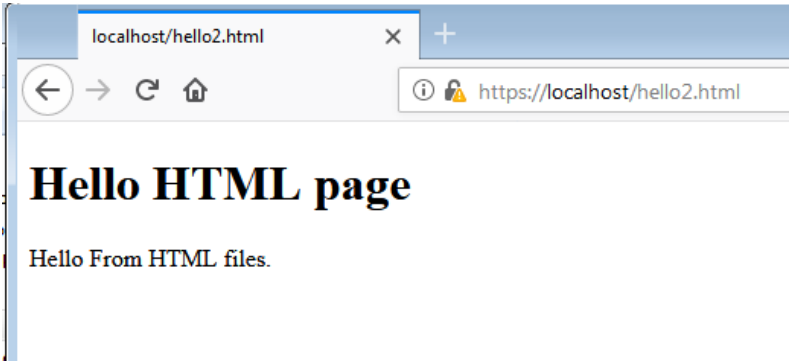
1. Buka Text Editor yang tersedia.
2. Simpan script #002 dengan nama hello2.html dan hello2.php

pada web server direktori. Untuk windows simpan di htdocs/Folder WWW web server. Linux /var/www.

3. Aktifkan Web Server.

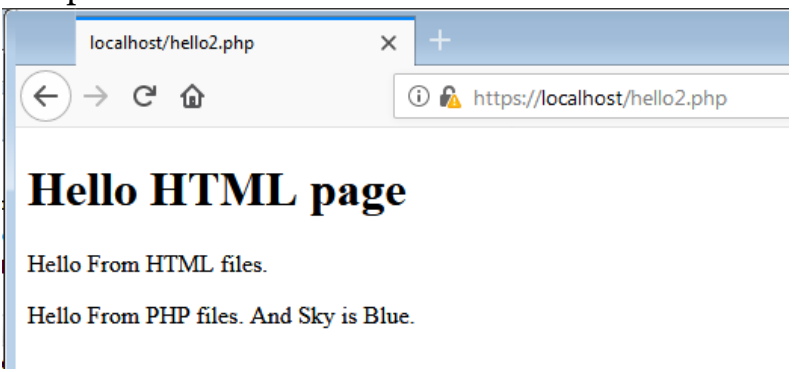
4. Buka web browser dengan alamat : https://localhost/hello2.html

Output:



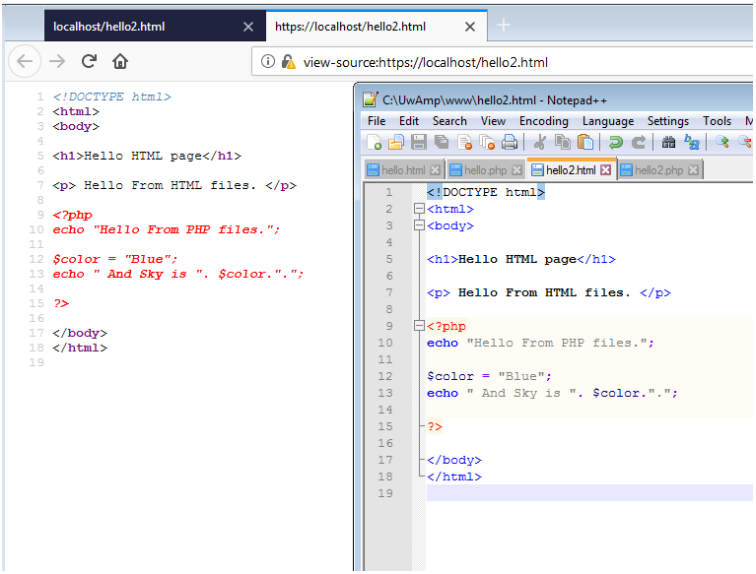
5. Buka tab baru dengan hello.html dengan web browser dengan alamat :https://localhost/hello2.php

Output:

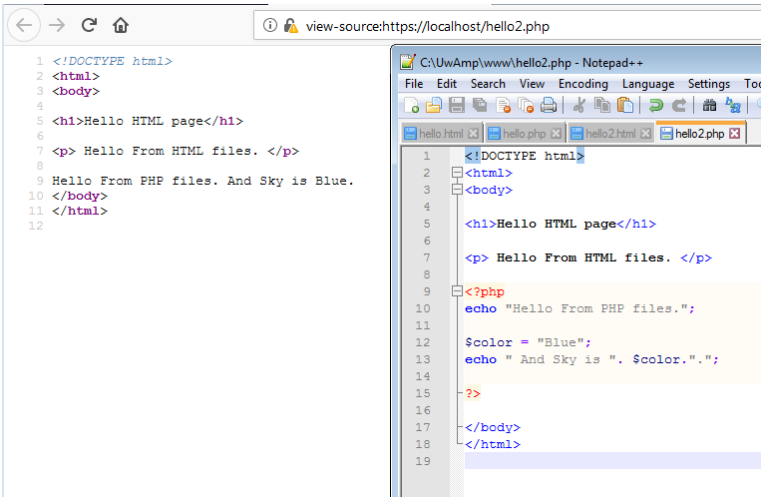


6. Klik kanan pada layar masing-masing tab pada langkah 4 dan 5. Dan pilih “View Page Source”. Amati kode yang ditampilkan. Jika pada hello2.html, akan ditampilkan semua kodenya termasuk kode php. Hal ini menunjukkan, kode php sangat berbahaya apabila ditaruh pada file HTML. Dan file php pada file HTMl tidak dijalankan oleh web server. Berbeda dengan kode php pada file .php. Dimana kode PHP di dalam tag `<?php ?>` tidak ditampilkan. Namun kode HTML dalam tag `<html> </html>` selain yang di dalam tag `<?php ?>` akan ditampilkan.

Output “View Page Source langkah” ke 4 (hello2.html):



Output “View Page Source langkah” ke 5 (hello2.php):



Soal dan Latihan Bab 1

1. Jelaskan perbedaan aplikasi Desktop, Mobile dan Berbasis Web.
2. Jelaskan keunggulan aplikasi berbasis Desktop dan Mobile bila dibandingkan dengan aplikasi berbasis Web
3. Jelaskan keunggulan aplikasi berbasis Web bila dibandingkan dengan aplikasi berbasis Desktop dan Mobile
4. Jelaskan ciri khas dari Client Side Scripting dan Server Side Scripting
5. Jelaskan mengapa file php (server side scripting) berbahaya apabila disisipkan pada file HTML. (Kegiatan 1.3. Langkah ke 6,7).

Bab 2. Layanan Web (Web Service)

Capaian Pembelajaran Mata Kuliah:

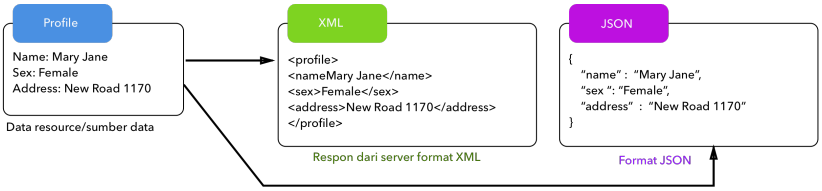
- Mahasiswa mampu mendeskripsikan peran Web Service dalam rangka pengembangan aplikasi web
- Mahasiswa mampu mendefinisikan SOA dan SOAP
- Mahasiswa mampu menjelaskan pemanfaatan file XML dan JSON dalam layanan Web.
- Mahasiswa mampu mengimplementasikan penggalian data pada sumber Web Service

Pada bab ini akan dijelaskan mengenai layanan Web atau yang biasa disebut Web Service. Web service merupakan istilah layanan yang diberikan kepada user/pengguna apabila pengguna ingin mengakses database. Dikarenakan akses langsung ke database sangatlah rentan terhadap aksi "kebocoran" data, maka peneliti memikirkan bagaimana memberikan akses database kepada pengguna/ mesin (sebagai pengguna) tanpa perlu direpotkan dengan pengaturan privileges. Yaitu pengaturan hak akses kepada device-device pengguna aplikasi. Terlebih, memberikan kepada hak akses perangkat seluler yang jumlahnya milyaran. Perlu diketahui, database yang dimaksud disini dapat berupa database tipe relasional atau database tipe non relational. Database tipe relasional biasanya disebut

database SQL. Sedangkan non relasional disebut database NoSQL.

Mungkin sebelumnya ada pertanyaan: Pada saat pengguna mengakses web service, apakah sebenarnya yang diakses? Pakai software apakah akses web service itu? Jawabnya, yang diakses adalah sebuah entitas data. Sebagai contoh pada suatu sistem kependudukan yang terdapat suatu entitas data penduduk yang terdiri dari Nama, Jenis Kelamin (Sex) dan alamat. Dimana secara kenyataannya data tersebut disimpan pada sebuah database. Web service berfungsi untuk menyediakan data tersebut dengan bentuk memberikan akses kepada sebuah file dalam bentuk sebuah halaman web yang agar dibaca oleh mesin/device lain. Jadi bisa dikatakan, web

service memberikan sebuah halaman web ketika ada “request” dari komputer client. File standard tersebut bisa berupa file XML dan File JSON. Contoh sebuah entitas/resource dan suatu penduduk, dapat ditransformasi menjadi sebuah file XML dan JSON seperti yang ditunjukkan pada gambar 2.1.



Gambar 2.1. Resource Entitas XML dan JSON

Tentunya agar komputer client tersebut dapat mengakses laman web yang dikatakan web service, bukan menggunakan sebuah aplikasi,

namun lebih tepatnya menggunakan suatu “standar” yang lebih dikenal dengan sebutan Protokol. Protokol tersebut yang sebagian pengguna telah menggunakannya. Yaitu Hypertext Transfer Protocol atau disingkat dengan HTTP. Jadi, inti dari adanya web service ialah bagaimana mengakses database (dan melakukan operasi kepada database seperti Create-Read-Update-Delete) hanya dengan melalui request melalui HTTP .

Nah jika web service itu memberikan akses dalam sebuah halaman web, yang menjadi pertanyaan selanjutnya adalah apakah laman tersebut merupakan halaman HTML atau sebuah akses ke file PHP, Python dan sebagainya? Jawabnya tidak. Untuk saat ini, dikenal dua tipe

Web Service yang menjadi topik penelitian bidang Rekayasa Perangkat Lunak paling banyak diimplementasikan. Yaitu Web Service Standard (WS-*) yang juga dikenal dengan Simple Object Access Protocol (SOAP) Web Services (menggunakan file format XML) dan Representational State Transfer (REST)/RESTful Web Service (menggunakan file format JSON).

Dalam literasi terkait Web Service, terdapat istilah SOA dan SOAP. Kedua istilah ini terlihat serupa, namun artinya sangat berbeda. SOA (Service Oriented Architecture) merupakan sebuah arsitektur. Artinya, SOA lebih kepada konsep/panduan untuk membangun sistem berbasis layanan (service). Sedangkan SOAP (Simple Object Access Protocol) lebih kepada

protokol yang mengimplementasikan dan memanfaatkan file berformat XML pada suatu aplikasi. Umumnya digunakan dalam aplikasi web berbasis layanan (Web Service). Karena konsep SOA merupakan "guidelines" dalam mengembangkan aplikasi berbasis layanan, maka prinsip SOA tidak harus diimplementasikan pada SOAP, dan penggunaan SOAP tidak selalu menerapkan konsep SOA. Penjelasan masing-masing web service ada pada Sub Bab 2.1 dan 2.2.

Beberapa hal lain mengapa sebuah aplikasi perlu untuk mengimplementasikan web service. Diantaranya:

1. Interoperabilitas

Dengan web service memungkinkan berbagai aplikasi untuk saling berkomunikasi dan berbagi (integrasi) data. Hal ini sangat berguna apabila beberapa perusahaan telah mengimplementasikan aplikasi yang beragam. Keberagaman sebagian besar disebabkan karena beberapa aplikasi yang telah dikembangkan dengan platform/bahasa pemrograman yang berbeda. Sebagai contoh, dalam perusahaan telah mengembangkan aplikasi penggajian menggunakan platform .NET, dan sistem absensi menggunakan PHP. Hal ini tidak jadi masalah apabila perusahaan ingin mengintegrasikan data pada kedua aplikasi tersebut. Yaitu mengembangkan web service pada kedua

aplikasi tersebut. Hal ini dapat dilakukan karena konsep web service adalah bagaimana dua mesin berkomunikasi menggunakan standar tertentu. Yaitu standar pada layer Transport, XML Messaging dan Service Discovery layer.

2. Cost Efficiency

Dikarenakan Web Service dapat diakses memanfaatkan protokol HTTP, aplikasi-aplikasi dapat menggunakan jaringan internet untuk integrasi datanya. Dimana lebih efisien (low bandwidth) apabila dibandingkan dengan solusi integrasi data pada layanan yang berlisensi (proprietary) seperti B2B EDI. Selain HTTP, Web Service juga dapat diimplementasikan pada

protokol lain yang lebih “mature” seperti File Transfer Protocol (FTP).

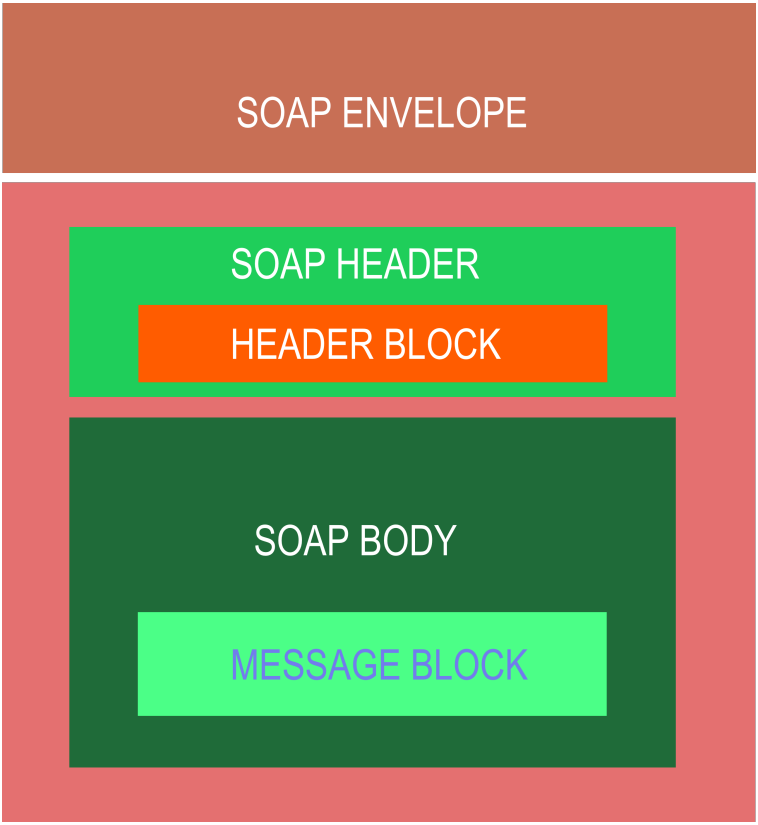
2.1. Simple Object Access Protocol

Integrasi data antar aplikasi sangat penting dalam untuk menghindari duplikasi, redundansi dan kebocoran data. Namun karena aplikasi-aplikasi dikembangkan dari platform/framework yang berbeda, integrasi data antara aplikasi heterogen ini akan menjadi sangat kompleks. Salah satu metode yang digunakan untuk memerangi kompleksitas ini adalah dengan menggunakan XML (Extensible Markup Language) sebagai “mediator” untuk bertukar data antara aplikasi yang berbeda platform tersebut.

Simple Object Access Protocol (SOAP) merupakan rekomendasi dari World Wide Web Consortium (W3C) untuk akses layanan Web menggunakan struktur bahasa WSDL (Web Services Description Language). Karena WSDL ini lebih kepada konsep yang digunakan untuk membangun sebuah web service, dan belum diimplementasikan menjadi suatu standar bahasa pemrograman (seperti bahasa HTML yang telah menjadi standar untuk membangun sebuah halaman Web), maka W3C merekomendasikan XML untuk mengimplementasikan WSDL. Untuk itu secara teori, apabila developer ingin mengembangkan aplikasi dengan memanfaatkan format XML, pengembang aplikasi memungkinkan mendesain

komunikasi antar aplikasi menggunakan beberapa protokol diluar protokol HTTP. Terlebih pada masa sekarang dimungkinkan beberapa aplikasi yang dibangun pada bahasa pemrograman yang berbeda. Setiap bahasa pemrograman dapat memahami bahasa markup XML. Oleh karena itu, XML digunakan sebagai bahasa baku/standar untuk pertukaran data. Akan tetapi XML pada awalnya tidak ada spesifikasi khusus terkait standar penggunaan XML untuk pertukaran data, maka SOAP dapat menjadi solusi atas permasalahan ini.

Arsitektur SOAP diilustrasikan sebagai berikut:



Gambar 2.3. Arsitektur Envelop pada SOAP

Struktur dari SOAP terdiri dari

1. Envelope

Sintaks pada elemen ini merupakan penanda bahwa blok pesan (message block) yang dikirim pada server ialah merupakan blok pesan yang mengimplementasikan XML.

2. Header

Sintaks pada elemen ini digunakan untuk menyimpan informasi terkait autentifikasi.

3. Body

Sintaks pada elemen ini digunakan untuk menyimpan isi dari data yang akan dikirim kepada server.

4. Fault

Sintaks ini digunakan untuk menyimpan informasi tambahan terkait blok pesan yang yang

dikirimkan. Seperti tipe dokumen yang dikirimkan, sukses atau tidak pesan terkirim kepada server.

Contoh Message Block pada SOAP ditunjukkan pada gambar 2.4. berikut (sumber: www.guru99.com):

Gambar 2.4. Contoh Envelop SOAP

Untuk SOAP Header ini sendiri terdiri dari 2 atribut. Yaitu :

1. Actor attribute

Dikarenakan packet pesan/data yang dikirimkan dari satu node ke node lain, Actor Attribute pada SOAP protocol mendefinisikan sebuah path/rute dari sender ke receiver. Untuk itu, dengan Actor Attribute, developer dapat menspesifikasikan penerima/end point dari paket/pesan/data yang dikirimkan. Contoh Actor Attribute ditunjukkan pada kode #2-001.

#2-001

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/
soap-envelope"
soap:encodingStyle="http://www.w3.org/
2001/12/soap-encoding">
<soap:Header>
<m:Trans xmlns:m="http://
www.w3schools.com/transaction/"
soap:actor="http://www.w3schools.com/
appml/">234
</m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

2. mustUnderstand Attribute

Atribut ini digunakan untuk mengindikasikan apakah entri header adalah wajib atau opsional bagi penerima untuk diproses. Jika disetting

mustUnderstand = "1", maka receiver/node selanjutnya yang akan yang memproses Header harus dapat membaca dan merekognisi elemen tersebut. Jika tidak dikenal, maka blok pesan tersebut tidak akan diproses oleh node tersebut. Contoh mustUnderstand Attribute:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/
soap-envelope"
soap:encodingStyle="http://www.w3.org/
2001/12/soap-encoding">
<soap:Header>
<m:Trans xmlns:m="http://
www.w3schools.com/transaction/"
soap:mustUnderstand="1">234
</m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

2.2. REST Web Service

Representational State Transfer (REST) di kenalkan oleh Roy Fielding tahun 2002 merupakan alternatif pengembangan web service dengan menggunakan JSON sebagai output dari layanan tersebut. REST Web Service menawarkan kemudahan dalam pengembangan web service. Kemudahan yang dimaksud salah satunya ialah mempermudah developer dalam melakukan integrasi data dengan melakukan operasi database melalui protokol HTTP. Apabila operasi database konvensional adalah Create, Read, Update, Delete, maka dalam sintaks REST API dikenal GET, PUT, DELETE. Tabel 2.1 menunjukkan perbandingan perintah/sintaks sql dan REST API.

Tabel 2.1. Mapping CRUD dan HTTP

CRUD Operation	HTTP Command	Input format	Output/Response Format
Create	POST	Encoded HTTP Form	Status 201 CREATED
Read	GET	URL Request Parameter	Determined by request headers
Update	PUT	Encoded HTTP Form	Status 200 OK
Delete	DELETE	URL Request Parameter	Status 200 OK

Soal dan Latihan Bab 2

1. Jelaskan peran Web Service dalam rangka pengembangan aplikasi web dan layanan Web?
2. Deskripsikan perbedaan SOA dan SOAP?
3. Deskripsikan perbedaan mendasar konsep SOA dan REST Web Service?
4. Jelaskan peran dan pemanfaatan file XML dan JSON dalam pengembangan layanan Web?
5. Implementasikan sebuah program sederhana dalam menggali data dari laman berikut: http://hepidad.github.io/data/top-250-ratings.json

Bab 3. Framework Pengembangan Aplikasi Web

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu menerangkan peran Framework dalam rangka pengembangan aplikasi web
- Mahasiswa mampu mengklasifikasikan Komponen Front-End Layer dan Back-End Layer
- Mahasiswa mampu menjelaskan peran pengembang Front-End Developer dan Back-End.
- Mahasiswa mampu mengimplementasikan Gaya Pengembangan Model View Controller.

Dalam mengembangkan layanan web service, perlu dikenalkan terlebih dahulu bagaimana mengembangkan aplikasi web. Karena web service sejatinya merupakan aplikasi web. Dalam mengembangkan aplikasi web, disarankan menggunakan sebuah kerangka kerja/framework yang membantu lebih cepat dalam dalam mengembangkan aplikasi web. Dalam pembahasan buku ini, digunakan framework berbasis bahasa pemrograman Python yang bernama Flask .

3.1. Flask Framework

Framework flask dikatakan sebuah Microframework karena ukuran dari framework tersebut yang relatif kecil.

Untuk memulai menggunakan Framework Flask, sebuah file Python butuh didefinisikan penggunaan Flasknya. Contoh pada sumber kode dibawah ini:

csv-read-web.py

```
1  from flask import Flask, render_template, request, redirect, url_for
2
3  app = Flask(__name__)
4  app.secret_key = 'xxxyyyzzz'
5
6
7  #Your Code Here
8
9
10
11 if __name__ == '__main__':
12     app.run(
13         host="0.0.0.0",
14         debug = True
15         )
16
```

Gambar 3.1. Contoh kode pada framework Flask

Pada baris ke 1, Modul Flask perlu diimpor beserta modul modul lainnya seperti modul “render_template”, request, redirect dan url_for.

Penjelasan masing-masing modul sebagai berikut:

1. Flask. Merupakan modul wajib yang diimpor apabila akan menggunakan Flask Framework.
2. Modul render_template. Merupakan modul yang digunakan untuk menampilkan file HTML yang terdapat pada direktori template.
3. Modul request. Merupakan modul untuk menangkap data yang dikirimkan oleh user dari sebuah Form HTML. Hal ini berguna pula untuk menangkap inputan dari user.
4. Modul redirect dan url_for. Digunakan untuk mengarahkan request yang diminta user ke sebuah method dalam file .py.

Penggunaan modul redirect biasanya sepasang dengan pemanfaatan modul url_for.

Untuk penjelasan lebih detail pemanfaatan dan penggunaan modul-modul diatas, dapat dilihat pada bagian 3.2.

Kegiatan 3.1.
Uji Coba Framework Flask

Alat dan Bahan:

Text Editor, Web Browser (Mozilla Firefox) dan Virtual Environment yang terinstall Flask

Petunjuk Uji Coba:

1. Buka Text Editor yang tersedia.
2. Buat sebuah direktori baru. Beri nama

direktori tersebut dengan nama helloFlask-Project. Dalam direktori tersebut, tulis kode program seperti pada gambar 3.1. Pada baris ke 7, tulis kode berikut:

```
@app.route('/')
def index():
    return "Hello from Flask!"
```

Simpan script dengan nama helloFlask.py

3. buat sebuah direktori baru dalam helloFlask-Project dengan nama templates.

4. Copy dan pindahkan file hello.html yang dihasilkan dari kegiatan 1.1 ke direktori templates. Ubah file hello.html sehingga menjadi seperti berikut:

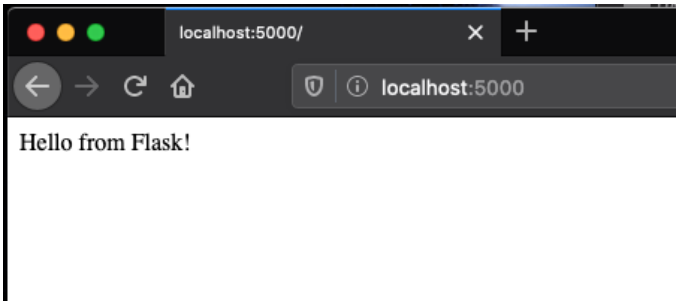
```
<!DOCTYPE html>
<html>
<body>

<h1>Hello HTML page</h1>

<p> Hello From HTML files. </p>

</body>
</html>
```

5. Buka kembali helloFlask.py. Tambahkan kode berikut:

```
@app.route('/hello')
def hello_page():
    return
render_template('hello.html')
```

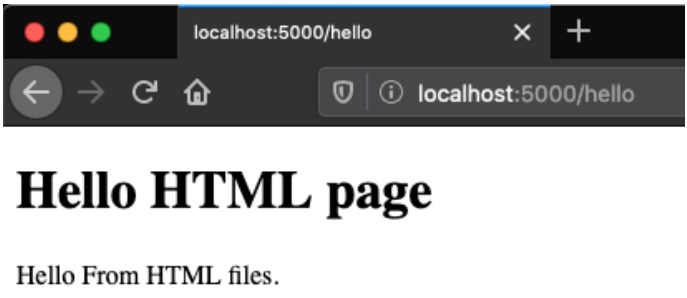
6. Kemudian coba dijalankan file helloFlask.py maka akan tampil laman berikut:

   a. Akses ke index (/): http://localhost:5000/ (Flask menggunakan port 5000 untuk developmentnya)



Gambar 3.2. Web Server aktif dari Flask

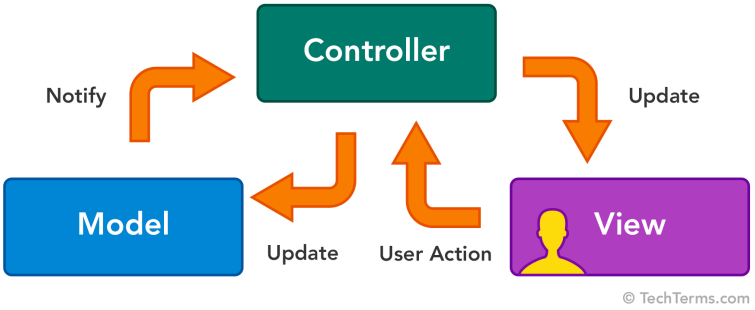
b. Akses ke laman hello (/hello): http://localhost:5000/hello



Gambar 3.3. Controller pada Flask

3.2. MVC

Dipilih Flask selain dari ukuran framework itu sendiri, Flask dipilih karena dapat mengakomodasi model pengembangan web dengan struktur MVC. Yaitu memisahkan unsur Model (Database), View (untuk Tampilan) dan Controller, untuk menangani request dari client. Ilustrasi fungsi dan tugas masing-masing unsur, digambarkan pada gambar 3.4.



Gambar 3.4. Konsep MVC pada Framework

Pada gambar diatas, dapat dilihat bahwa apa yang ditampilkan oleh user merupakan fungsi dari unsur View. Namun ketika terjadi interaksi (dalam bentuk request ke halaman web tertentu) oleh pengguna, yang meng-handling ialah unsur controller. Apabila, bentuk request dari user tersebut ialah memanggil suatu data maka unsur model ini berperan.

Pada kegiatan 3.1. dapat dilihat sebagai contoh kecil penggunaan MVC. Untuk, controller didefinisikan pada script berikut:

```
@app.route('/')
def index():
    return “Hello from Flask!”

@app.route('/hello')
def hello_page():
    return
render_template('hello.html')
```

Dari script diatas, "@app.route()" melakukan kontrol dengan membaca "request" dari user. Apabila user melakukan request URI berikut: "http://localhost:5000/hello", maka secara otomatis Flask akan me-reroute request ke method yang dibawah "@app.route(/hello)", yaitu menampilkan halaman HTML (hello.html). Modul "render_template" pada object return ke file HTML, merepresentasikan implementasi dari "View" layer dari konsep MVC.

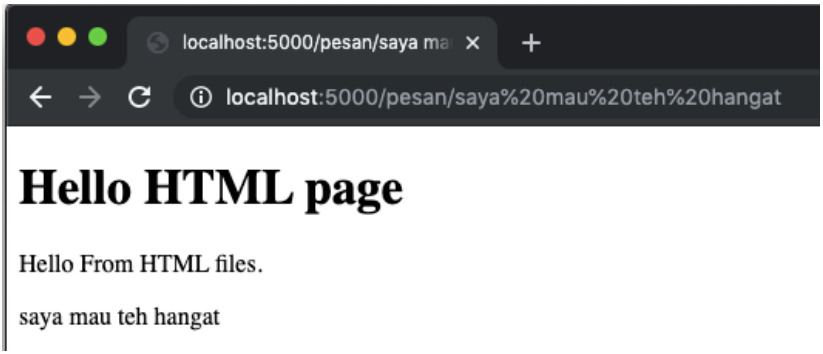
Di dalam Flask juga dapat memparsing sebuah "variable" langsung dari perhitungan dari script utama. Sebagai contoh perhatikan kode berikut:

```
app.route('/pesan/<var_pesan>')
def hello_pesan():
    return
render_template('hello.html',pesan
=var_pesan)
```

Lalu edit hello.html menjadi seperti kode berikut:

```
<!DOCTYPE html>

<html>

<body>

<h1>Hello HTML page</h1>

<p> Hello From HTML files. </p>

<p>{{pesan}}</p>

</body>

</html>
```

Selanjutnya diaktifkan helloFlask.py nya. Kemudian akses ke “http://localhost:5000/pesan/saya mau teh hangat”, maka oleh browser URL akan di konversi menjadi “http://localhost:5000/pesan/saya%20mau%20teh%20hangat” dan setiap sintaks setelah /pesan/ akan dimunculkan pada tag `<p>{{pesan}}</>` , seperti pada tampilan berikut:



Gambar 3.5. Variable Parsing pada Flask

Lalu bagaimana bentuk real implementasi dari Framework MVC ini pada transaksi yang terkoneksi pada sebuah basis data/bentuk Model layernya? Hal ini akan dijelaskan secara detail pada sub bab 3.3.

3.3. CRUD

Dalam penguasaan skill dalam membangun sebuah aplikasi web, pada dasarnya developer cukup memiliki kemampuan melakukan CRUD pada framework yang dipilih. CRUD yang dimaksud disini adalah operasi Create, Read, Update dan Delete pada suatu record database. Dan untuk mengimplementasikan Model pada konsep MVC dan interaksi, kita mengambil suatu contoh interaksi data sebuah mobil. Terdapat suatu data mobil seperti Tabel 3.1. Untuk melayani transaksi dan menampilkan data tersebut, maka perlu membuat sebuah Form Inputan dan Form View dalam format HTML untuk melakukan

entry (Create,Insert) dan menampilkan data yang telah di entry ke database tersebut.

Table 3.1. Contoh Data Brand Mobil dan Harganya

ID	Brand	Type	Price
111	Toyota	Hiace	35000
222	Honda	Accord	28000
333	Suzuki	Katana	15000

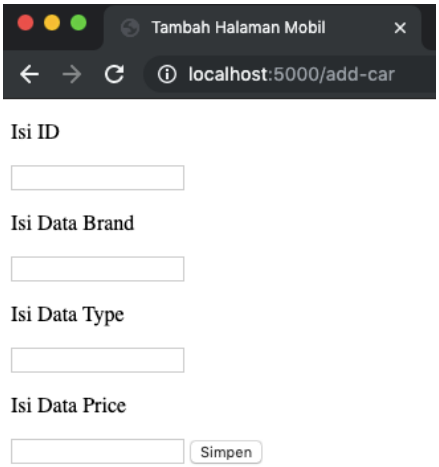
Form inputan sederhana dalam HTML dapat menggunakan standar tag `<form> </form>`. Selain form inputan, maka perlu pula tampilan HTML untuk menampilkan data yang telah diinput user. Contoh form inputan dan tampilan dalam tabel ditunjukkan pada script berikut.

```
<!DOCTYPE html>
<html>
<head>
    <title>
        Tambah Halaman Mobil
    </title>
</head>

<body>

<form action = ""  method="POST" >
    <p>Isi ID</p>
    <input type="text" name="dataID">

    <p>Isi Data Brand</p>
    <input type="text" name="dataBrand">

    <p>Isi Data Type</p>
    <input type="text" name="dataType">

    <p>Isi Data Price</p>
    <input type="text" name="dataPrice">
        <button>Simpen</button>
</form>
<p>{{pesan}}</p>
</body>

</html>
```

Setelah disimpan dengan nama addCar.html, maka selanjutnya kita buat controller baru agar dapat diakses oleh user. Yaitu dengan menambahkan kode berikut:

```
@app.route('/add-car')
def add_car():
    return
render_template('addCar.html')
```

Selanjutnya user dapat mengakses melalui URL /add-car.



Gambar 3.6. Form input pada endpoint /add-car

Sampai disini, kita hanya menampilkan inputan saja kepada user. Selanjutnya kita perlu mengolah inputan tersebut agar masuk ke database. Untuk lebih mudahnya, database yang digunakan ialah yang berbasis SQL dan relational. Dalam buku ajar ini menggunakan modul peewee ORM untuk relational tabelnya.

Dan menggunakan SQlite sebagai database engine. Perlu dikerahui bahwa apabila menggunakan SQLite, database yang digunakan merupakan database berbasis file bukan service. Sehingga mahasiswa yang baru mengenal permrograman databae tidak perlu direpotkan untuk menginstall database engine berbasis service. Seperti MariaDb/mysql. Untuk menggunakan peewee orm, diperlukan instalasi peewee orm flask:

```
pip install peewee
```

Dan mendifinisikan penggunaan peewee dengan import modulnya sebagai berikut:

```
import sqlite3 as lite
from peewee import *
```

Untuk mendedinisikan tabel dalam aplikasi yang dikembangkan menggunakan flask framework, tabel di definisikan sebegai berikut:

```
DATABASE = car.db
database = SqliteDatabase(DATABASE)

class BaseModel(Model):
    class Meta:
        database = database

class Cars(BaseModel):
    carID = CharField(unique=True)
    brand = TextField()
   type = TextField()
   price = TextField()

def create_tables():
```

```
        with database:
database.create_tables([Cars])
```

Kemudian saat program dijalankan, perlu dieksekusi perintah create table sebelum main modul Flask dijalankan.

Gambar 3.7. Aktivasi create_tables()

Jika sebelumnya database belum dibuat, maka secara otomatis akan membuat file database baru.

Selanjutnya yang diperlukan ialah bagaimana menangkap inputan dari user untuk mengisi data tersebut. Disinilah peran HTML dan modul request dari python.

Untuk menangkap inputan dari Form HTML, menggunakan modul request dari Flask. Untuk penggunaanya mengikuti kaidah berikut:

```
varInputan = request.form['name']
```

Sehingga apabila dalam file HTML seperti berikut:

```
<input type="text" name="dataID">
```

Maka `['name']` perlu dirubah sesuai dengan "name" di form HTMLnya. Dalam contoh ini, kaidahnya menjadi :

```
varInputan = request.form['dataID']
```

Untuk varInputan itu sendiri, digunakan untuk menyimpan variable yang diinput user. Variabel yang tersimpan di varInputan inilah yang nantinya dapat disimpan dalam database.

Untuk menyimpan di database, jika menggunakan peewee ORM, kita hanya perlu memanggil tabel yang sudah kita definisikan di bagian model.

```
@app.route('/add-car-save',methods=['POST'])
def add_car_save():
    carID_form = request.form['dataID']
    carBrand_form = request.form['dataBrand']
    carType_form = request.form['dataType']
    carPrice_form = request.form['dataPrice']
    car_entry = TabeLog.create(
            carID = carID_form,
            carBrand = carBrand_form,
            carType =  carType_form,
            carPrice = carPrice_form
            )

    return "Saved!"
```

Gambar 3.8. Script untuk simpan database

Soal dan Latihan Bab 3

1. Jelaskan peran Framework dalam rangka pengembangan aplikasi web?

2. Jelaskan komponen Front-End Layer dan Back-End Layer?

3. Jelaskan peran pengembang Front-End Developer dan Back-End?

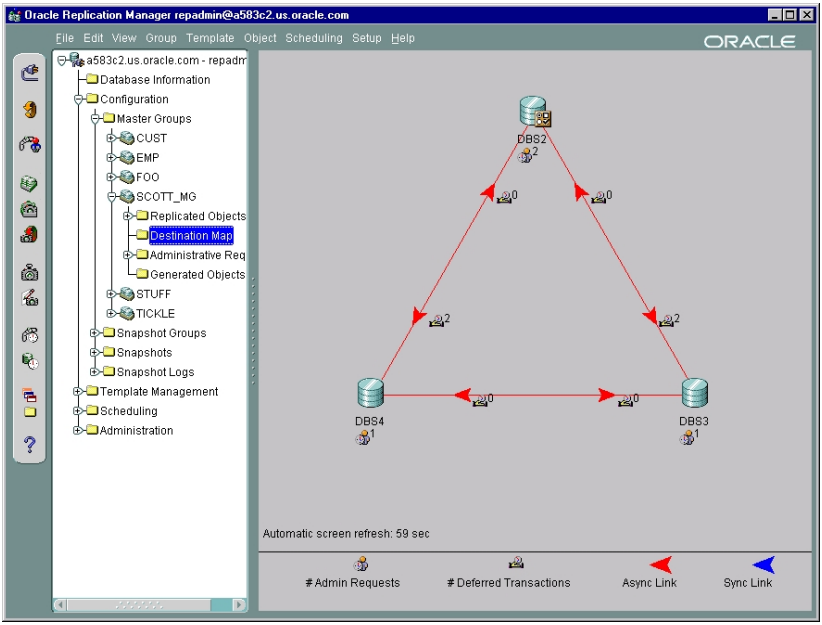
4. Implementasikan pemrograman database menggunakan kerangka Model View Controller.

Bab 4. Microframework dan Microservice

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu menerangkan peran Micro Framework dalam pengembangan teknologi web.
- Mahasiswa mampu menjelaskan fungsi routing pada teknologi Web yang ada pada saat ini.
- Mahasiswa mampu memanfaatkan mesin template untuk renderisasi file HTML.
- Mahasiswa mampu megembangkan aksi Create Read Update dan Delete pada halaman Hypertext Markup Language (HTML) dan mesin Database.

Integrasi data dapat dilakukan secara otomatis antar mesin. Diantaranya integrasi pada teknologi eksisting dengan memanfaatkan fitur dari dari Database Engine itu sendiri. Fitur yang dimaksud ialah fitur replikasi database. Replikasi database yang dilakukan ialah dalam rangka untuk melakukan backup/sinkronisasi data pada suatu transaksi atau pada waktu tertentu. Sebagai contoh fitur replikasi dari Oracle Database yang dapat dilakukan secara periodik atau by request. Fitur replikasi pada Oracle Database inilah yang menjadikan alasan mengapa database Oracle banyak digunakan. Contoh fitur replikasi ditunjukkan pada gambar 4.1. berikut (sumber: oracle.com)



Gambar 4.1. Fitur Replikasi pada Database Oracle

Kemudian bagaimana dengan Database Engine yang tidak memiliki fitur tersebut? Terdapat alternatif lain apabila kita menggunakan database yang minimalis seperti

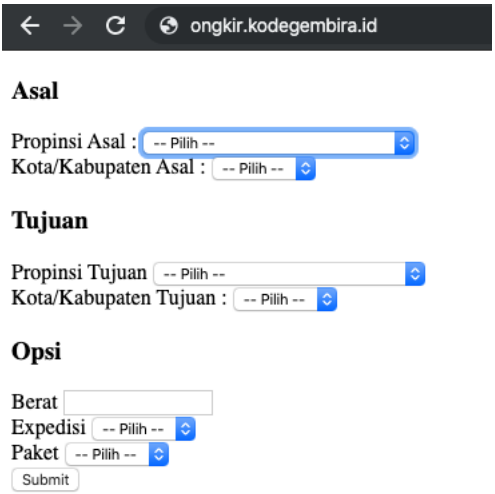
SQLite yang tidak memiliki fitur seperti di oracle. Yaitu dengan memanfaatkan web service.

Pada Bab 3, dijelaskan interaksi suatu aplikasi yang secara langsung berinteraksi dengan database. Berbeda dengan bab sebelumnya, pada bab ini, akan dijelaskan bagaimana melakukan CRUD dengan tidak berinteraksi langsung dengan database, namun melalui sebuah REST API. Untuk penjelasan REST API pada bab 4.1.

4.1. REST API

Pada ecommerce, biasanya dibutuhkan sebuah perhitungan nilai ongkos kirim. Karena banyaknya penyedia ongkos kirim, dan harga ongkos kirim sewaktu-waktu berubah, maka jasa

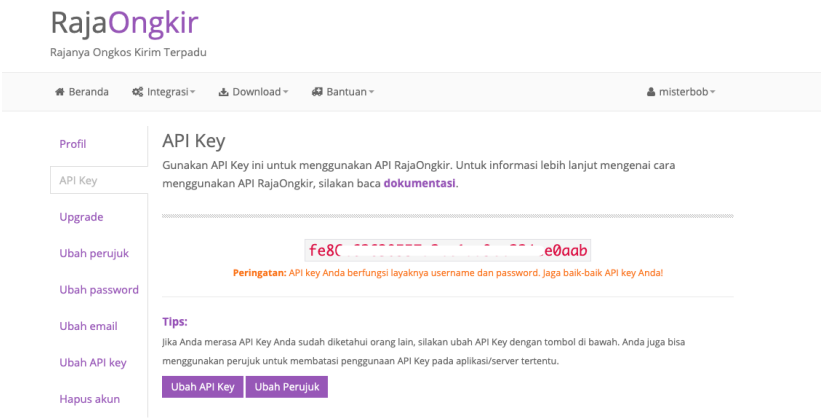
layanan ongkos kirim tersebut harus menyediakan informasi terupdate terkait besaran jasa ongkos kirimnya. Contoh sebuah perhitungan jasa perhitungan ongkos kirim dengan memanfaatkan REST API dapat diakses pada alamat http://ongkir.kodegembira.id/.



Gambar 4.2. REST API untuk cek ongkos kirim

Pada laman diatas, perhitungan jasa ongkos kirim tidaklah dilakukan pada mesin server http://ongkir.kodegembira.id/. Melainkan menggunakan REST API dari penyedia informasi jasa pengiriman yang dapat digunakan secara gratis dan berbayar. Sebagai contoh layanan informasi jasa pengiriman yang ada di rajaongkir.com (ROC). Untuk mencoba layanan REST API dari ROC, perlu sebuah token/API key yang diperoleh setelah mendaftar di ROC. Setelah mendaftar, dan melakukan aktivasi, API KEY dapat diakses melalui menu akun kemudi pilih sub menu panel. Token/API Key yang disediakan pada awal mulanya ialah tipe starter untuk akses layanan yang gratis. Namun pengguna dapat mengupgrade tipe akun menjadi

Basic dan Pro. Dimana perbedaannya ialah di kelengkapan fitur yang dapat diakses.



Gambar 4.3. Token/API Key

Untuk mencoba akses REST API pada ROC, kita dapat menggunakan aplikasi command promt/terminal bawaan dari sistem operasi. dengan mengetikkan perintah yang berisi parameter-parameter yang dibutuhkan untuk mengakses layanan yang disediakan oleh ROC

tersebut. Sebagai contoh layanan untuk mencari ongkos kirim barang dari kota A ke Kota B. Untuk mendapatkan informasi besaran nila jasa ongkos kirim membutuhkan beberapa parameter diantaranya yaitu:

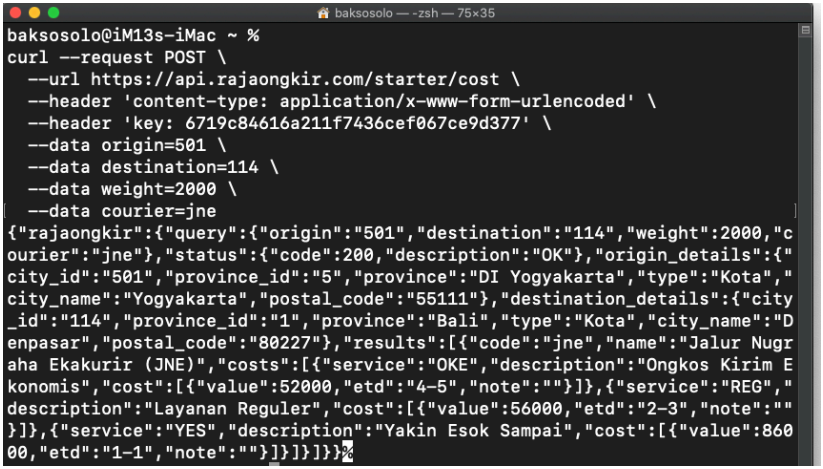
Tabel 4.1. Paramater REST API

Method	Parameter	Wajib	Tipe	Keterangan
POST/ HEAD	key	Ya	String	API Key
POST/ HEAD	android-key	Tidak	String	Identitas aplikasi Android
POST/ HEAD	ios-key	Tidak	String	Identitas aplikasi iOS
POST	origin	Ya	String	ID kota atau kabupaten asal
POST	destination	Ya	String	ID kota atau kabupaten tujuan
POST	weight	Ya	Int	Berat kiriman dalam gram
POST	courier	Ya	String	Kode kurir: jne, pos, tiki.

Dengan melihat tabel parameter diatas, terdapat parameter yang wajib diberikan dan ada yang tidak wajib. Wajib disini artinya perlu di definisikan. Maka melihat tabel diatas, kita perlu mendefinisikan parameter key,origin,destination,weight dan courier. Maka pada terminal kita perlu definisikan parameter dalam perintah berikut:

```
curl --request POST \
  --url https://api.rajaongkir.com/
starter/cost \
  --header 'content-type: application/
x-www-form-urlencoded' \
  --header 'key: API-Key-kita' \
  --data origin=501 \
  --data destination=114 \
  --data weight=1700 \
  --data courier=jne
```

Maka akan ditampilkan hasil dari “request” sebagai berisi informasi (dalam bentuk JSON) berisi informasi nilai cost dan informasi terkait nama paket jasa yang pada courier yang dipilih.



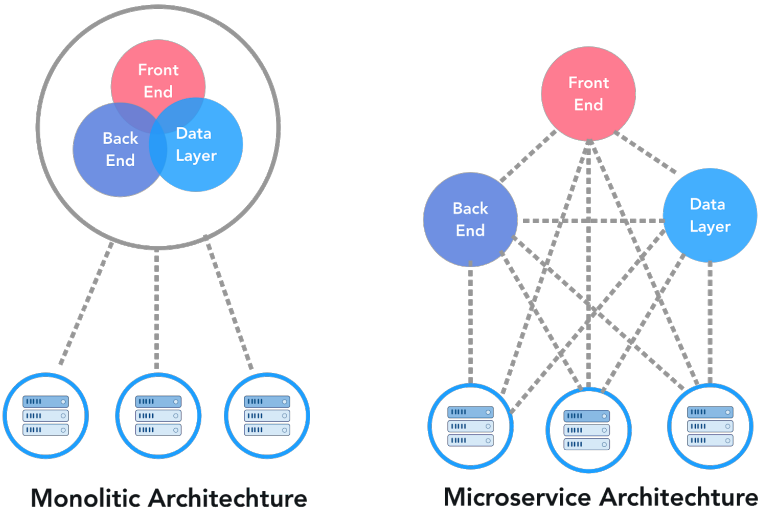
Gambar 4.4. Contoh Akses REST Web Service

Dari gambar diatas dapat dilihat besaran jasa ongkos kirim dari kota dengan kode origin=501 dan destination=114.

4.2. Microservice

Melihat banyaknya pengembangan modul-modul pada suatu aplikasi yang berkembang. Tim pengembang dipastikan akan menghadapi masalah-masalah terkait skalabilitas platform yang telah dikembangkan. Dimana, pada saat ini, umumny aplikasi yang telah dikembangkan merupakan aplikasi Monolithic. Artinya, semua layanan (request) pada saat transaksi dilayani oleh aplikasi tunggal dan tidak didistribusikan menjadi beberapa aplikasi kecil berbasis web service [6]. Tentunya hal ini akan berdampak pada performa aplikasi yang terdiri dari banyak modul transaksi dalam satu platform [7]. Sebagai solusi, pengembangan platform yang

akan dikembangkan secara modular menggunakan arsitektur Microservice.



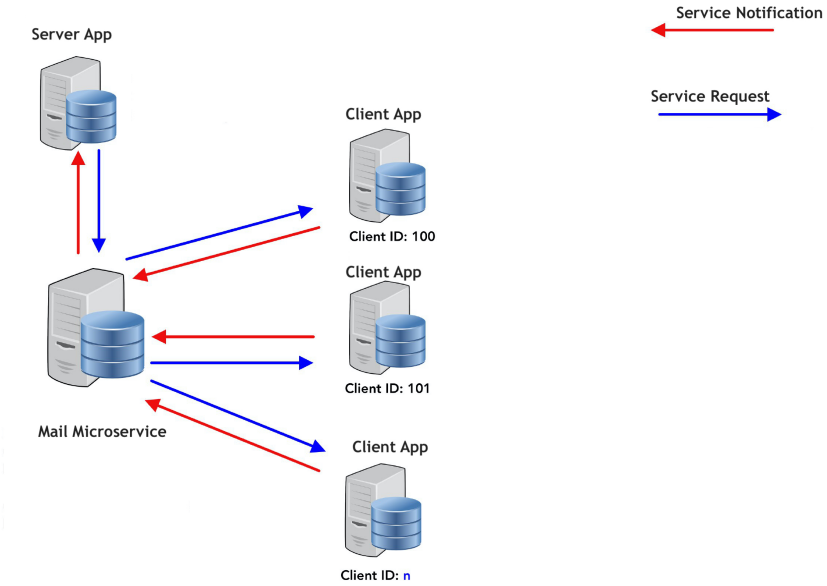
Gambar 4.5. Arsitektur Monolithic dan Microservice

Nilai lebih dalam pemanfaatan arsitektur Microservice adalah tim pengembang dapat membangun aplikasi-aplikasi yang akan dikembangan menjadi sebuah modul-modul kecil

[8]. Dimana, modul-modul kecil aplikasi tersebut saling berkomunikasi (data transfer, GET, POST) satu sama lain melalui web service [9]. Untuk itu, pengembangan aplikasi dengan model arsitektur Microservice memungkinkan mendistribusikan kompleksitas proses pengembangan aplikasi kepada beberapa kelompok tim-tim pengembangan yang lebih kecil [10]. Model pengembangan aplikasi Microservice tersebut sangat berpotensi untuk dikerjakan oleh beberapa kelompok mahasiswa sebagai studi kasus/penerapan ilmu serta kolaborasi antara mahasiswa. Dimana masing-masing kelompok tersebut akan mengembangkan aplikasi secara modular.

4.3. Implementasi Microservice: Layanan Email

Untuk contoh pengembangan microservice pada buku ajar ini ialah mengembangkan microservice pengiriman email.



Gambar 4.6. Arsitektur Email Microservice

Dalam implementasi pengembangan aplikasi web, biasanya terdapat suatu proses registrasi/ pendaftaran email. Jika tanpa arsitektur microservice (monolitic) , notifikasi email dilakukan oleh main program setelah main program menjalankan proses registrasi.

Implementasi awal microservice dapat dilakukan dengan memisahkan proses kirim notifikasi menjadi program tersendiri. Dimana eksekusi notifikasi via email diletakkan pada service yang independen. Untuk itu kita mencoba membuat dua aplikasi independen. Sebut saja server-mail dan client-mail.

Server-mail berfungsi merespon permintaan notifikasi via email dari client-mail. Sehingga beban eksekusi pada client-mail terkurangi

karena beban eksekusi program notifikasi sudah di-“handle” oleh service/aplikasi lain. Dalam hal ini dikendalikan oleh server-mail. Untuk client-mail kita simpan dengan nama client-mailms.py. Sedangkan server-mail disimpan dengan nama server-mailms.py.

Seperti yang telah dijelaskan sebelumnya, pada awal penggunaan Flask Microframework, perlu didefinisikan modul-modul yang akan digunakan pada pengembangan server-mail dan client-mail. Pada client-mail dan server-mail sama-sama membutuhkan modul-modul yang tertulis pada kode #4-001. Yaitu, modul requests, json, time, sqlite, peewee, Flask, dan flask-mail. Untuk lebih jelasnya silahkan dicermati kode #4-001 berikut.

#4-001

```
import requests, json, time, random, os
import sqlite3 as lite
from peewee import *
from flask import Flask, render_template,
request, redirect, url_for, session,
jsonify
from flask_mail import Mail, Message
```

Kemudian kita perlu mendefinisikan database dan file configurasi lainnya. Seperti yang ditunjukkan pada kode #4-002

#4-002

```
app = Flask(__name__)
mail = Mail(app)

DATABASE = "YOUR_DB.db"
database = SqliteDatabase(DATABASE)

from config import Config
hostconf=file('host.cfg')
hostcfg=Config(hostconf)
```

Untuk DATABASE = “YOUR_DB.db” disesuaikan dengan jenis server-mail dan client-mail. Database ini digunakan untuk menyimpan inputan/email yang telah diterima/berhasil dikirimkan pada microservice tersebut. Untuk client-mail dapat menggunakan nama clientmailms.db. Untuk server-mail menggunakan servermailms.db.

Selanjutnya kita perlu mendeviniskan Model atau yang dalam istilah framework MVC ialah model databsenya. Dalam FLask Microframeowrk, mendefiniskan database/Model cukup membuat class-clas tiap model. Karena ini ujicoba microservice, kita cukup memiliki satu tabel saja. Untuk mendefinisikan tabel tersebut, ditunjukkan oleh kode #4-003.

#4-003

```
class BaseModel(Model):
    class Meta:
        database = database

class TBEmail(BaseModel):
    tbemail_id =
BigAutoField(unique=True)
    email_sent_from = TextField()
    email_recipient = TextField()
    email_subject = TextField()
    email_content = TextField()
    email_time_epoch = TextField()
    email_time_stamp = TextField()

def create_tables():
    with database:
        database.create_tables([TBEmail])
```

Setelah mendefinisikan tabel sebagai database yang akan digunakan, kita perlu mendefinisikan konfigurasi standar ketika server dijalankan. Untuk ini biasanya kita memasukkan fungsi/metode “create_tables()” dan konfigurasi

nomor port. Seperti yang tertulis pada kode #4-004.

#4-004

```
if __name__ == '__main__':
    create_tables()
    app.run(
        debug=True,
        host="0.0.0.0",
        port=5022
    )
```

Untuk server-mail mungkin tetap menggunakan port 5000. Sehingga tidak perlu menulis port=5022. Sedangkan untuk client, tetap mendefinisikan port yang digunakan ialah port=5022. Dan yang membedakan client-mail dan server-mail lainnya yaitu untuk akses dilakukan oleh client-mail saja. Sehingga modul “requests” diimplementasikan pada client-

mailms.py. Seperti yang ditunjukkan pada kode #4-007.

#4-005

```
@app.route('/')
def index():
    return render_template(‘index-
mail.html')

@app.route('/
sent_mail',methods=['GET','POST'])
def sent_mail():
    email_recipient =
request.form['recipient']
    email_subject =
request.form['subject']
    email_content =
request.form['content']
```

Kode diatas (#4-005) merupakan kode untuk menangka inputan isian dari kode HTML. Yang disimpan dengan nama index-html. HTML ini digunakan untuk ujicoba kirim email dengan

inputan recipient, subject dan content. Index-mail.html ditunjukkan pada kode #4-006 berikut:

#4-006

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport"
content="width=device-width, initial-
scale=1">
    <title>Yet Another Mail Client -
Challenge#6</title>
    <meta name="description"
content="Source code generated using
layoutit.com">
    <meta name="author"
content="LayoutIt!">
    <link href="../static/css/
bootstrap.min.css" rel="stylesheet">
    <link href="../static/css/style.css"
rel="stylesheet"></head>
```

```
<body>
    <div class="container-fluid">
    <div class="row">
    <div class="col-md-12">
    <h3>Sent email to your friends!</h3>
<form role="form"
action="{{url_for('sent_mail')}}"
method="POST">
<div class="form-group">
<label for="recipient">
 Email Address:
 </label>
 <input type="email" name="recipient"
class="form-control" id="recipient">
</div>
<div class="form-group">
<label for="subject"> Subject    </label>
<input type="text" name="subject"
class="form-control" id="subject">
 </div>
<div class="form-group purple-border-
focus">
<label for="content">Message </label>
<textarea id="content" name="content"
rows="6" style="width:100%;"> </textarea>
</div>
<button  type="submit" class="btn btn-
primary">Sent</button>
</form></div></div></div>
</body>
</html>
```

Langkah terakhir dari pengembangan client-mail ini ialah implementasi “request” ke server-mail. request tersebut terletak pada kode #4-007.

#4-007

```
@app.route('/
sent_mail',methods=['GET','POST'])
def sent_mail():
    email_recipient =
request.form['recipient']
    email_subject =
request.form['subject']
    email_content =
request.form['content']
datas = {
        "recipient":email_recipient,
        "subject": email_subject,
        "content": email_content
        }
    url=hostcfg.urlserver
    headers = {'Content-type':
'application/json', 'Accept': 'text/
plain'}
    r = requests.get(url,
data=json.dumps(datas), headers=headers)
    return r.text
```

Pada kode #4-007, request dilakukan ketika sudah berhasil meng-collect variable dari inputan HMTL.

Yang menjadi pertanyaan. request oleh variabel “r” ini melakukan request ke mana? Jawabnya, ke server-mail dimana alamat URLnya tertulis pada file yang telah terkonfigurasi pada kode #4-002. Yaitu “host.cfg”.

Untuk kode pada server-mail, berbeda dengan kode client-mail. Dimana server-mail tidak melakukan “request”. Akan tetapi menerima “request”. Bentuk/definis program menerima “request” yaitu dengan membuat class abc(Resource). Artinya, jika terdapat suatu class abc(Resource) pada kode tersebut, maka

terdapat suatu Web Service abc yang dapat memberikan respon apabila datang suatu “request” dengan parameter tertentu. Contoh class Resource ditunjukkan pada kode #4-008.

#4-008

```
class MS(Resource):
    def get(self):
        parserMS =
reqparse.RequestParser()

       parserMS.add_argument('recipient')
       parserMS.add_argument('subject')
       parserMS.add_argument(‘content')

       argsMS = parserMS.parse_args()

        email_recipient=
argsMS.get('recipient')
        email_subject=
argsMS.get('subject')
        email_content=
argsMS.get('content')
```

```
datas = {
        "recipient":email_recipient,
        "subject": email_subject,
        "content": email_content

        return jsonify(datas)
```

Bila dibandingkan kode #4-007 dan #4-008, terdapat perbedaan yang cukup signifikan. Yaitu return dari #4-007 merupakan text yang di dapat ketika request ke server-mail. Sedangkan pada #4-008, return ke json variabel “datas”. Nah, variabel “datas” inilah yang ditangkap client-mail pada statemen kode “return r.text()”.

Soal dan Latihan Bab 4

1. Jelaskan perbedaan REST API dan SOAP dari sisi file standard yang digunakan.

2. Jelaskan perbedaan arsitektur Monlitic dan Microservice?

3. Jelaskan unsur utama dalam pengembangan aplikasi client-server yang menggunakan arsitektur Microservice?

Bab 5. Keamanan Layanan Web Service Security

- Capaian Pembelajaran Mata Kuliah:
- Mahasiswa mampu menunjukkan celah-celah (holes) pada teknologi web.
- Mahasiswa mampu menunjukkan celah-celah (holes) pada layanan web.
- Mahasiswa mampu menjelaskan fungsi Tokenisasi pada layanan Web.
- Mahasiswa mampu memanfaatkan modul/API dalam pengamanan layanan Web.

5.1. Tokenization

Memberikan layanan web/web service seperti memberikan akses/jalan pintas kepada publik untuk mengakses/merubah data tanpa berurusan dengan database. Untuk itu, akses ke sebuah web service yang sedang dikembangkan perlu untuk diperketat keamanan tersebut dengan membuat suatu token pada layanan tersebut.

Token berfungsi sebagai alat autentifikasi layaknya user/password. Apabila user dan pasword merupakan autentifikasi yang digunakan oleh manusia sebagai user [11,14]. Token merupakan autentifikasi yang digunakan oleh sebuah mesin ketika melakukan “request” pada suatu web service yang dikembangkan.

Pemanfaatan token biasanya disediakan oleh sebuah API Provider dalam rangka membedakan antara user subcription. Seperti yang digambarkan pada gambar 4.3. Token juga digunakan untuk autentifikasi kode promo dalam suatu eccomerce. Maka belajar membuat token perlu dilakukan oleh rekan mahasiswa.

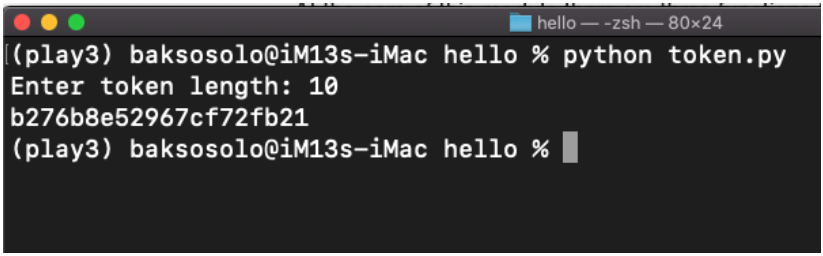
Untuk membuat token sendiri caranya cukup mudah. Yaitu dengan menggunakan modul secret pada python 3.x. Seperti yang dicontohkan sebagai gambar 5.1. berikut.

```
1  import secrets
2
3  #Enter token length.
4  token_length=input("Enter token length: ")
5
6  print(secrets.token_hex(int(token_length)))
```

Gambar 5.1. Kode Pembuatan Token

Penjelasan dari kode pada gambar 5.1. adalah sebagai berikut:

1. Baris ke 1 melakukan impor modul secrets.
2. Baris ke 4 memberikan inputan pada user berapa panjang token yang akan dibuat. Modul secret akan membuat token sebanya n x 2 karakter. Artinya apabila user memasukkan panjang token 10, maka panjang token yang dibuat ialah 20 karakter.
3. Baris ke 6 menampilkan token yang dibuat.

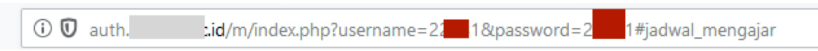


Gambar 5.2. Pembuatan Token

5.2. URL Safe Serialization

Dalam membuat web service, developer disarankan untuk memperhatikan desain URL atau alamat service yang terekspose pada public [12]. Karena URL yang menggunakan parameter integer biasa seperti :"http:/suatuweb.com/data?id=911" akan mudah untuk digali atau dicrawling oleh pihak ketiga. Karena parameter yang tertera pada URL "id=911" mengindikasikan bahwa terdapat suatu id lain yang terdiri dari angka numerik lainnya. Misalnya ada "id=912" dan seterusnya. Terlebih penulis menemukan sebuah celah keamanan pada suatu sistem informasi. Yang berasal dari sebuah dokumen petunjuk penggunaan aplikasi tersebut pada suatu institusi. Celah keamanan

tersebut dikarenakan ketidakhati-hatian dari developer dalam mendesain suatu URL ketika mengembangkan aplikasi berbasis web tersebut. Berikut contoh kesalahan tersebut:



Gambar 5.3. Celah Keamaanan pada URL

Bisa dilihat dari gambar diatas, URL yang diakses menampilkan variabel user dan password. Hal ini sangatlah berbahaya apabila diimplementasikan pada aplikasi yang sarat data sensitif. Misalnya sistem informasi perbankan dan kesehatan. Untuk menghindari hal tersebut, pengembang perlu mengkonversi key yang ditampilkan pada URL menjadi deretan string

yang tertokenisasi. Atau yang biasa disebut URL Serialization.

Untuk implementasi pada Flask Microframework, hal ini sangatlah mudah karena dapat menggunakan modul URL Safe Serialization dari paket ItsDangerous (https://itsdangerous.palletsprojects.com/en/1.1.x/url_safe/). Untuk cara penggunaannya dapat dicontohkan pada Gambar. Untuk penjelesan tiap baris adalah sebagai berikut:

1. Baris 1 merupakan definisi penggunaan paket itsdangerous dan import modul URLSafeTimedSerializer.
2. Baris ke 3 dan 4 mendefinisikan sebuah variabel s merupakan variabel

URLSafeTimedSerializer dengan kata kunci ‘writesomethingsecrethere’. Kata kunci ini harus dirahasiakan.

urlsafe.py

```
1  from itsdangerous import  URLSafeTimedSerializer
2
3  global s
4  s = URLSafeTimedSerializer('writesomethingsecrethere')
5
6
7  #Enter some key that need to be secret.
8  input_key=raw_input("Enter some key: ")
9
10 token = s.dumps(input_key, salt="applejuice")
11
12 print "From key :",input_key,"convert to:",token
13
14 #Enter some token for reveale the secret.
15 input_token=raw_input("Enter some token: ")
16 key = s.loads(input_token, salt="applejuice")
17
18 print "From :",input_token,"convert to key:",key
19
```

Gambar 5.4. Modul URL Safe Serialization

3. Baris ke 8 merupakan instruksi untuk menerima inputan dari user yang kita anggap nanti menjadi suatu key di URL dan mau

dirahasiakan. Key dapat berupa sebuah kata atau angka numerik.

4. Baris ke 10 merupakan proses URL Serialization, dimana merubah key pada point 3 sebelumnya menjadi sebuah deret bilangan dan huruf yang tidak dapat ditebak maksudnya.

5. Baris ke 12 menampilkan hasil perubahan key ke URL Serialization.

6. Baris ke 15 merupakan instruksi untuk menerima inputan berupa key yang telah di URL Serialization menjadi suatu key yang kita input pada point ke 3.

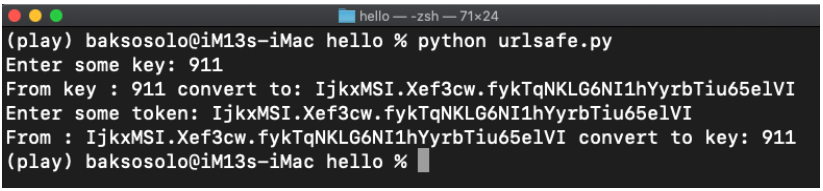
7. Baris ke 16 merubah URL Serialization menjadi key awal. Perubahan dari URL

Serialization menjadi ke Key awal agar selanjutnya Key dapat digunakan sebagai keyword pada statemen SQL dan sebagainya.

8. Baris ke 18 menampilkan hasil perubahan URL Serialization ke Key awal.

Point 1 hingga 8 menunjukkan key dapat dikonversi menjadi suatu deret string panjang dan membuat URL tidak mudah ditebak. Sehingga URL yang terdapat celah keamanan seperti “http:/suatuweb.com/data?id=911” menjadi http:/suatuweb.com/data?id=IjkxMSI.Xef3cw.fykTqNKLG6NI1hYyrbTiu65elVI”. Diaman deretan string “IjkxMSI.Xef3cw.fykTqNKLG6NI1hYyrbTiu65elVI” merupaka URL Serialization dari angka 911.

Sebagaimana ditunjukkan pada gambar xxx. Yang merupakan hasil eksekusi dari kode program pada gambar xx.



Gambar 5.5. Eksekusi URL Safe Serialization

Soal dan Latihan Bab 5

1. Jelaskan kegunaan token pada layanan web service?

2. Jelaskan fungsi pengaman URL pada layanan Web Service?

3. Implementasikan pembuatan token dengan 40 karakter.

4. Impelementasikan pembuatan URL yang aman untuk key = 007.

Bab 6. Implementasi Web Service: Teknologi Web API

Capaian Pembelajaran Mata Kuliah:

- Mahasiswa mampu menjelaskan peran API pada layanan Web
- Mahasiswa mampu melakukan operasi data menggunakan API melalui protokol HTTP dan halaman HTML.

6.1. API Marketplace

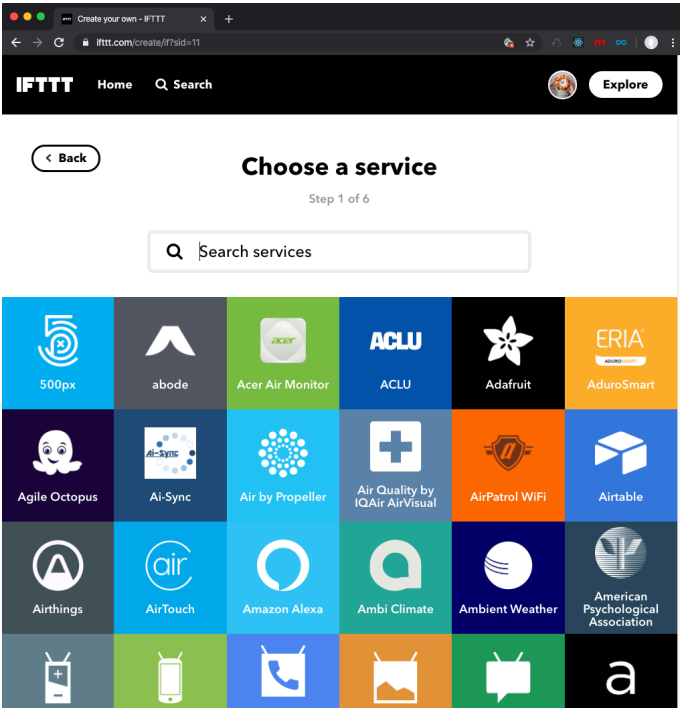
Sebelum membahas API marketplace, rekan mahasiswa coba menjawab pertanyaan berikut? "Apakah perbedaan ecommerce Bhinneka.com dan Bukalapak.com?" Jawabnya, Bhinneka.com merupakan murni ecommerce. Bukan marketplace. Karena ecommerce pada Bhinneka.com tidak menyediakan publik untuk "berjualan" pada eccomerce tersebut. Sedangkan Bukalapak.com merupakan eccomerce dan marketplace karena memang menyediakan tempat (place) pada public sehingga terciptanya sebuah pasar (market). Lalu bagaimana dengan API marketplace? Seperti Bukalapak.com, API Marketplace merupakan sebuah sebutan untuk eccomerce yang memiliki market, dimana publik

dapat pula berjualan di ecommerce tersebut. Namun di API Marketplace, yang dijual adalah API. API marketplace yang ada saat ini sangat banyak. Sebagaimana terlist pada laman https://www.g2.com/categories/api-marketplace.

Untuk contoh penggunaan API marketplace pada buku ajar ini menggunakan layanan pada IFTTT (https://ifttt.com/). Dengan skenario menyimpan catatan singkat dengan mempostingan pada twitter dengan tagar (hashtag) #simpanini. Selanjutnya catatan tersebut ditampilkan dalam laman web.

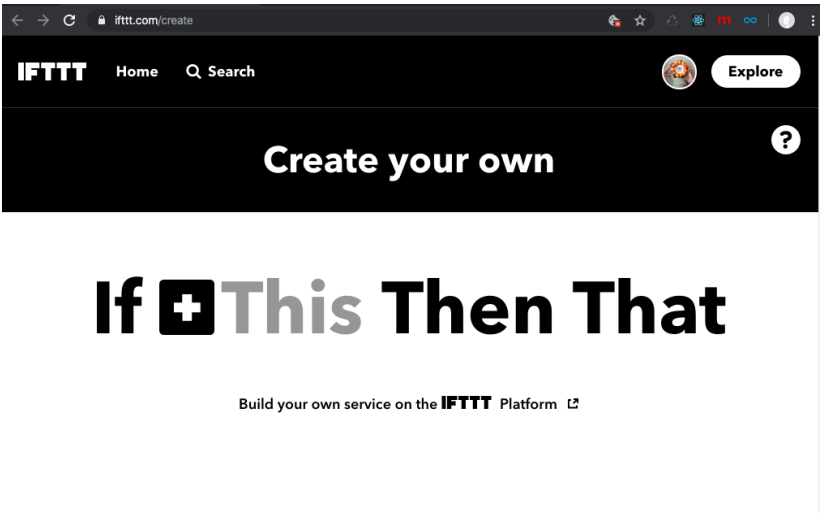
Untuk memulai project diatas, maka rekan mahasiswa perlu memperhatikan langkah berikut:

1. Membuat akun di IFTTT.com.



Gambar 6.1. Dahsboard API Marketplace

2. Membuat layanan baru dengan klik profil dan sub menu create.

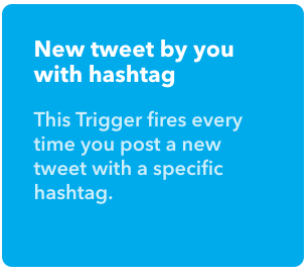


Gambar 6.2. Service baru pada API Marketplace

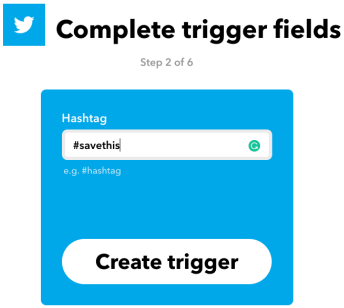
3. Klik kata “This” untuk memilih layanan yang menjadi “trigger”. Trigger disini maksudnya ialah suatu event/aksi yang nantinya akan menjalankan perintah berikutnya. Even ini macamnya banyak. Namun yang kita butuhkan disini event yang terjadi apabila terdapat suatu

hastag pada twitter, maka kita perlu pilih twitter untuk layanan yang menjadi trigger-nya.

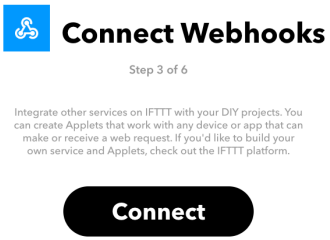
4. Pilih trigger ketika ada hastag.



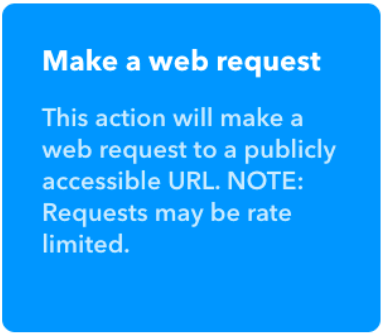
5. Pilih hastag yang menjadi trigger. Dalam contoh ini hastag #savethis.



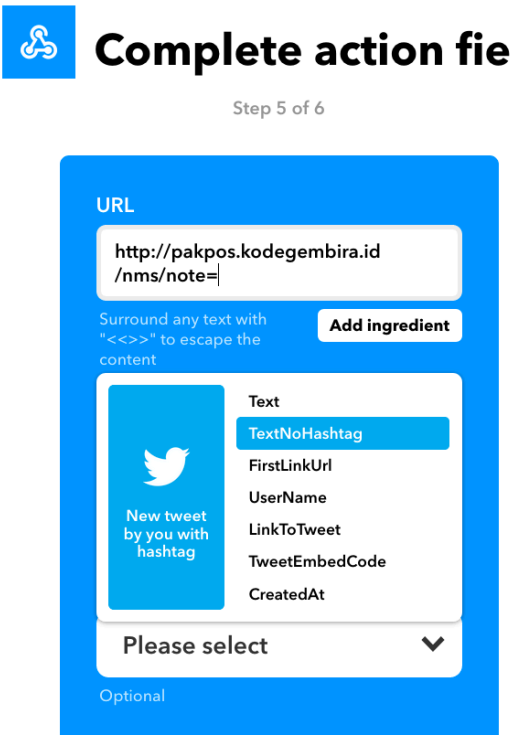
6. Selanjutnya pilih reaksi dari trigger tersebut. Yaitu akses ke sebuah web aplikasi yang dibuat. Untuk itu pilih reaksi berupa webhooks.



7. Untuk aksi di webhooks sendiri hanya ada satu yaitu “Make a web request”.



8. Kemudian kita perlu setting URL, Method dan Content Type. Untuk URL diisikan: http://pakpos.kodegembira.id/nms/note={{TextNoHashtag}}, dan Methode GET serta Content Type ke text/plain.



Gambar 6.3. Trigger pada API Marketplace

Sampai disini, setting web service dari IFTTT API marketplace sudah lengkap. Selanjutnya kita perlu membuat sebuah web aplikasi dengan Flask Microframework untuk web service yang nantinya akan menampilkan “catatan” dalam laman web yang kita miliki dengan kode #6-001.

Pada kode #6-001 menggunakan modul Flask-RESTAPI. Dimana kita perlu mendifinisikan sebuah resource Web Service pada sebuah class Resource. Bentuk umumnya ialah `class nama_class(resources)`. Dan untuk penentuan alamat dari resource ini yaitu `api.add_resource(nama_class, '/alamat_akses/', endpoint='alamat_akses/')`.

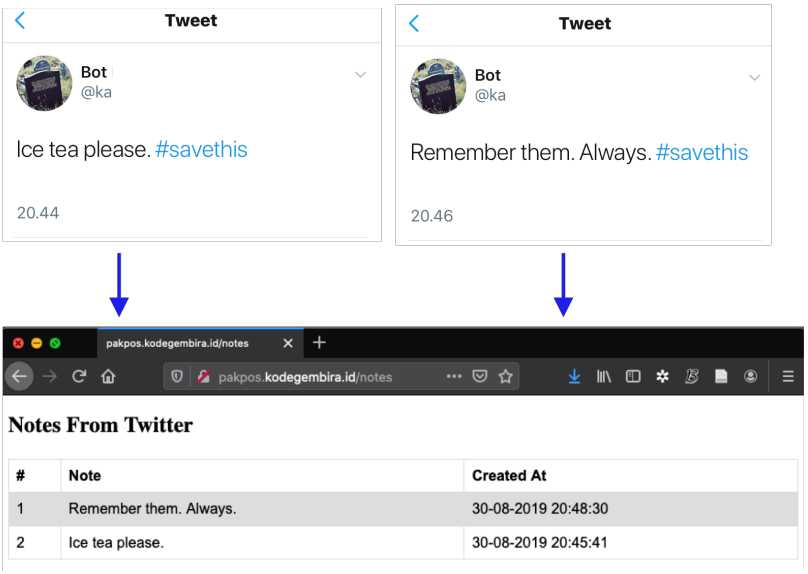
#6-001

```
import json, requests, time, os

import sqlite3 as lite
from peewee import *

from flask import Flask, render_template,
request, redirect, url_for, session,
jsonify

from flask_restful import Resource, Api,
reqparse

app = Flask(__name__)
api = Api(app)

DATABASE = "note.db"
database = SqliteDatabase(DATABASE)

class BaseModel(Model):
    class Meta:
        database = database

class TBNote(BaseModel):
    note_id = BigAutoField(unique=True)
    note = TextField()
    note_epoch = TextField()
    note_time = TextField()
```

```
@app.route('/notes')
def notes():
    rows =
TBNote().select().order_by(TBNote.note_id
.desc())
    return render_template('notes.html',
rows=rows)

class NoteMS(Resource):
    def get(self):
        parserMS =
reqparse.RequestParser()
        parserMS.add_argument('note')

        argsMS = parserMS.parse_args()
        noteData=argsMS.get('note')
        note_epoch_Data = time.time()
        note_time_Data =
str(time.strftime('%d-%m-%Y %H:%M:%S',
time.localtime(note_epoch_Data)))

note_save_db(noteData,note_epoch_Data,not
e_time_Data)

        datas = {
        'status': '000',
        'note': noteData,
        "note_time": note_time_Data
        }

        return jsonify(datas)
```

```
def note_save_db(x,y,z):
    email_entry = TBNote.create(
            note=x,
            note_epoch= y,
            note_time= z
            )
    return True

def create_tables():
    with database:
        database.create_tables([TBNote])

api.add_resource(NoteMS, '/nms/',
endpoint='nms/')

app.secret_key="noteAPImarketplace"

if __name__ == '__main__':
    create_tables()
    app.run(
        debug=True,
        host="0.0.0.0"
    )
```

Untuk ujicoba API tersebut, kita perlu membuat suatu posting pada twitter, dengan tagar #savethis. Kemudian, text pada postingan tersebut akan muncul pada laman yang telah kita siapkan. Yaitu pada laman "/notes".



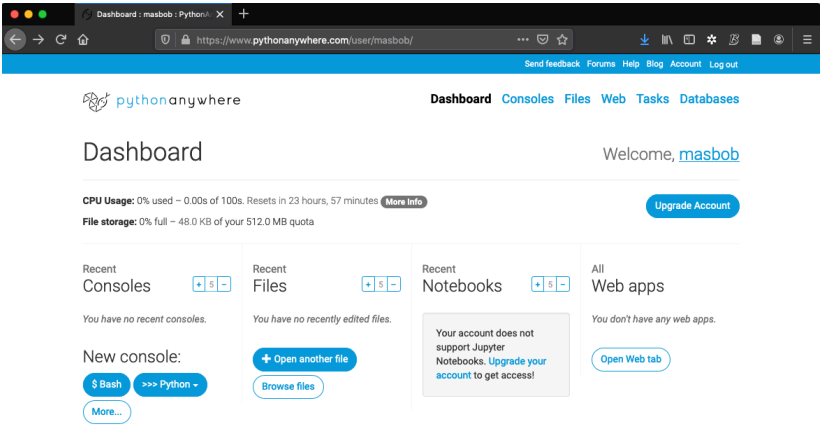
Gambar 6.4. Ujicoba trigger pada API Marketplace

6.2. API Provider

Setelah kita mencoba suatu API pada marketplace, saatnya kita mencoba menjadi API Provider. Dimana membangun suatu aplikasi web yang menyediakan API untuk pengguna.

Untuk ujicoba membangun aplikasi WEB yang menjadi API Provider disini akan dicontohkan mengembangkan API Provider yang menyediakan layanan notifikasi via email. Sehingga pengguna API dapat mengirimkan email kepada clientnya dengan menggunakan layanan API yang kita bangun. Untuk menjadi API provider, tentunya kita harus menyediakan suatu laman web yang aktif selama 24/7. Untuk hal tersebut kita dapat menggunakan layanan hosting dan domain gratis yang support layanan

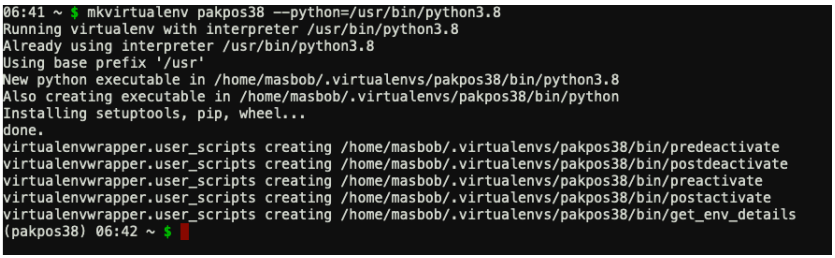
Flask MicroFramework yaitu PythonAnywhere.com.



Gambar 6.5. Platform untuk API Provider

Setelah rekan mahasiswa register dan login, silahkan menuju menu web. Klik tombol add a new web app. Pilih Framework Flask serta Python 3.8.

Sebelum memulai project, disarankan untukmembuat virtual environtment sendiri dimana kita bebas menambahkan modul-modul yang dibutuhkan dalam pengembangan aplikasi ini. Untuk membuat virtual environtment, kita klik tombol console, lalu ketikkan perintah: mkvirtualenv pakpos38 —python=/usr/bin/python3.8.



Gambar 6.6. Console VE untuk API Provider

Selanjutnya yang perlu diinstall adalah peewee, Flask, flask-restful, flask-mail dan config. Untuk lebih mudahnya, membuat file

requirements.txt berikut dan melakukan perintah install dengan pip install -r requirements.txt.

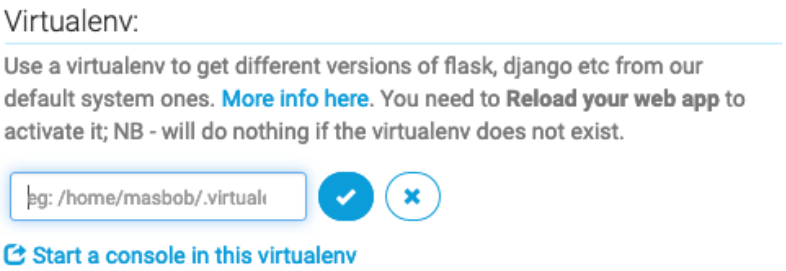
#6-002

```
aniso8601==8.0.0
blinker==1.4
Click==7.0
config==0.4.2
Flask==1.1.1
Flask-Mail==0.9.1
flask-peewee==3.0.3
Flask-RESTful==0.3.7
itsdangerous==1.1.0
Jinja2==2.10.3
MarkupSafe==1.1.1
peewee==3.12.0
pytz==2019.3
six==1.13.0
Werkzeug==0.16.0
wtf-peewee==3.0.0
WTForms==2.2.1
```

Apabila virtualenvirontment telah dibuat, maka label paling kiri pada console akan tertulis

nama virtual environtment yang dibuat sebelumnya. Dalam hal ini ialah pakpos38. Yaitu menggunakan python 3.8.

Langkah selanjutnya, kita perlu mendefiniskan virtualenvirontment yang dipilih ialah pakpos38. Dengan cara kembali ke menu web dan input pakpos38 pada isian Virtualenv. Seperti yang diilustrasikan pada gambar 6.7.

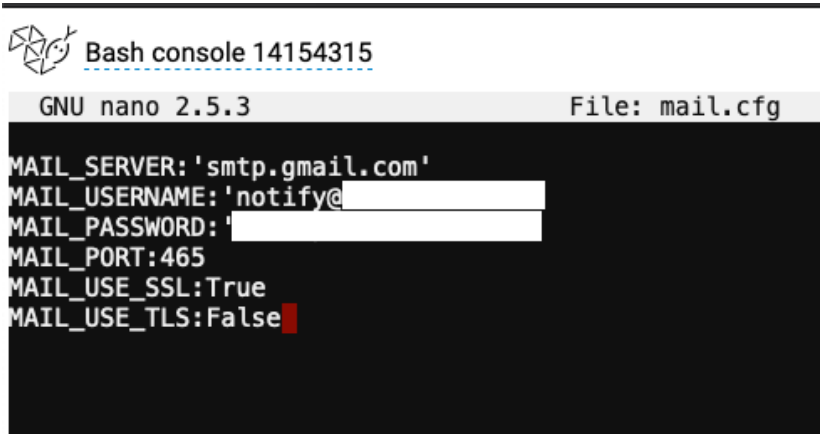


Gambar 6.7. Virtual Environtment untuk API Provider

Karena kita akan membuat layanan API Provider pengiriman email, maka kita perlu

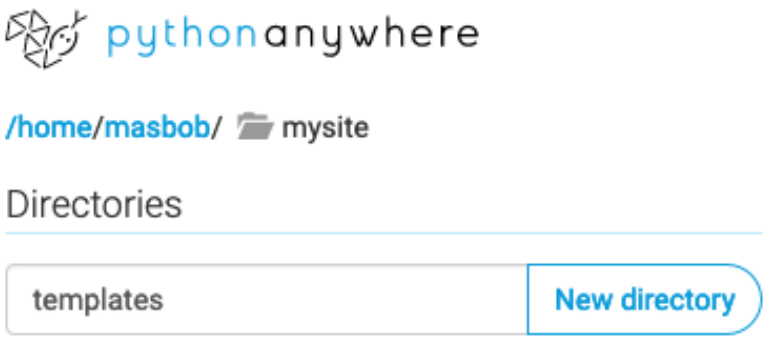
definisikan konfigurasi email kita. Deinisi tersebut dengan membuat sebuah file baru didalam direktori project yang bernama mail.cfg. Untuk mengirim email melalui aplikasi lain menggunakan akun google mail, kita perlu menuliskan parameter MAIL_SERVER, MAIL_USERNAME, MAIL_PASSWORD, MAIL_PORT, MAIL_USE SSL, MAIL_USE_TLS. Keenam parameter tersebut perlu diisi sesuai gambar 6.8.

Khusus untuk MAIL_USERNAME dan MAIL_PASSWORD, disesuaikan dengan user dan password akun Google Mail. Selanjutnya dibuat laman HTML untuk input/kirim email.



Gambar 6.8. File .cfg untuk API Provider

Seperti yang telah dijelaskan pada bab sebelumnya. Dalam Flask Microframework file-file HTML diletakkan pada direktori template/. Maka kita perlu membuat direktori template pada dashboard PythonAnywhere. Dengan cara pilih menu Files, kemudian pada isian Directories diisikan “templates” seperti yang dicontohkan pada gambar 6.9.



Gambar 6.9. Setting direktori templates

Dengan cara yang sama, dan kali ini masuk ke direktori templates yang sudah dibuat. Serta dipastikan keterangan PATH pada dashboard sudah mengarah ke /home/masbob/mysite/templates. Kemudian kita perlu membuat laman HTML untuk input/kirim email. Yaitu dengan menyiman kode pada direktori template/kirimemail.html. kirimemail.html ditunjukkan pada kode #6-003 berikut:

#6-003

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport"
content="width=device-width, initial-
scale=1">

    <title>PakPos - Yet Another Mail Client
</title>

    <meta name="description" content="Source
code generated using layoutit.com">
    <meta name="author" content="LayoutIt!">

    <link href="../static/css/
bootstrap.min.css" rel="stylesheet">
    <link href="../static/css/style.css"
rel="stylesheet">

  </head>
  <body>
    <br>

    <div class="container-fluid">
    <div class="row">
        <div class="col-md-12">
```

```
            <h3>
                Sent email to your friends!
            </h3>

            <p>
                Made with ☕ by @hepidad
            </p>

            <form role="form"
action="{{url_for('do_kirim_email')}}"
method="POST">
                <div class="form-group">

                    <label for="recipient">
                        Email Address:
                    </label>
                    <input type="email"
name="recipient" class="form-control"
id="recipient">
                </div>
                <div class="form-group">

                    <label for="subject">
                        Subject
                    </label>
                    <input type="text"
name="subject" class="form-control"
id="subject">
                </div>

                <div class="form-group
purple-border-focus">
                    <label for="content">
                        Message
                    </label>
                    <textarea id="content"
name="content" rows="6" style="width:100%;">
</textarea>
                </div>
```

```
                <button  type="submit"
class="btn btn-primary">
                    Sent
                </button>
            </form>
        </div>
    </div>
</div>

    <script src="../static/js/
jquery.min.js"></script>
    <script src="../static/js/
bootstrap.min.js"></script>
    <script src="../static/js/
scripts.js"></script>
  </body>
</html>
```

Setelah mendefinisikan file inputan HTML, kita perlu mendefinisikan REST API yang nantinya akan menjadi API Provider. Melihat inputan paramater file HTML (kirimemail.html) untuk ujicoba kirim email, membutuhkan setidaknya 3 parameter. Yaitu Recipient, Subject dan Message Contentnya. Maka kita perlu mendefinsikan ada 3 parser/paramater pada API

Provider yang dibuat. Definisi tersebut ditunjukkan pada class MailAPI kode #6-004.

#6-004

```
class MailAPI(Resource):
    def get(self):
        parserMS = reqparse.RequestParser()

        parserMS.add_argument('recipient')
        parserMS.add_argument('subject')
        parserMS.add_argument('content')

        argsMS = parserMS.parse_args()

        email_recipient=
argsMS.get('recipient')
        email_subject=
argsMS.get('subject')
        email_content=
argsMS.get(‘content')

        kirimemail()

        stat={
        'status': ‘000’,
        "recipient":email_recipient,
        "subject": email_subject,
        "content": email_content
        }

       return jsonify(stat)
```

Yang terpenting dalam membuat API Provider ini ialah mendefinisikan class pada script sebelumnya pada aplikasi yang dibuat dengan cara menambahkan baris kode #6-005 berikut:

#6-005

```
api.add_resource(MailAPI, '/kirim-email-
do/', endpoint='kirim-email-do/')
```

Sampai disini kita sudah membuat sebuah API Provider. Hal terakhir yang dilakukan ialah membuat sebuah metode untuk mengirim email yang tangkap melalu parser ke email yang dituju melalui SMTP Client. Yaitu membuat metode kirimemail() yang diperlukan oleh kode #6-004.

Mengirim email melalui SMTP Client, dapat menggunakan modul Flask-Mail. Sebagaimana ditulis pada kode #6-006 berikut:

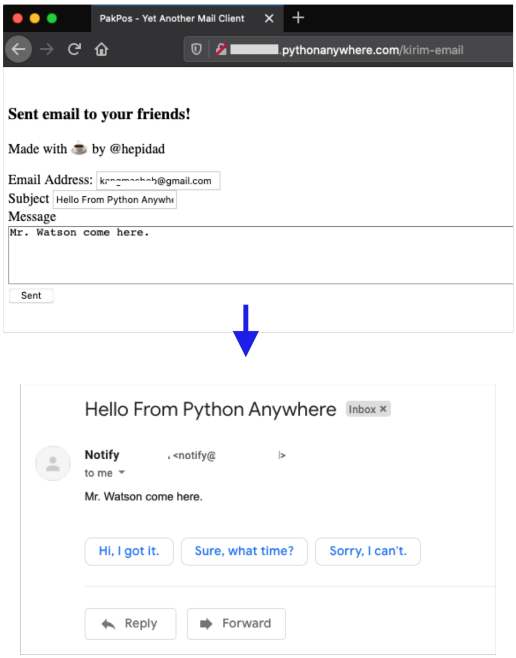
#6-006

Setelah melakukan “reload” pada dashboard PythonAnywhere, kite dapat melakukan ujicoba

```
from flask_mail import Mail, Message

app = Flask(__name__)
app.config.from_pyfile('mail.cfg')
mail = Mail(app)

def kirimemail(recipient,subject,content):
    msg = Message(subject,
sender='notify@emailanda.com',
recipients=[recipient])

    msg.body = content

    mail.send(msg)

    return True
```

API Provider untuk layanan kirim email dengan

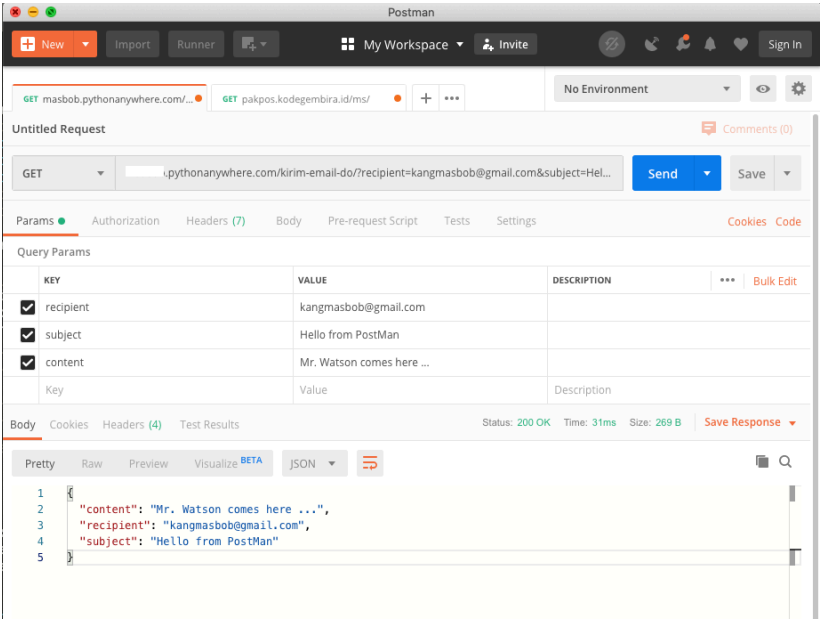
dua cara. Pertama yaitu melakukan request langsung pada "/kirim-email". Dan yang kedua ialah melalui aplikasi PostMan.



Gambar 6.10. Uji API Provider via "/kirim-email"

Untuk ujicoba API Provide pada Postman, kita perlu mendefinisikan URL dan 3 parameter

yang dibutuhkan yaitu recipient, subject dan content. Input URL dan parameter pada PostMan ditunjukkan pada gambar 6.12 berikut.



Gambar 6.11. Uji API Provider pada PostMan

Soal dan Latihan Bab 6

1. Jelaskan perbedaan Marketplace dan API Marketplace?

2. Jelaskan manfaat API Marketplace dalam pengembangan aplikasi/rekayasa perangkat lunak?

3. Membuat Project menggunakan IFTTT dengan menangkap twitter yang mention akun rekan mahasiswa.

4. Menampilkan postingan pada twitter yang mention akun rekan mahasaiswa pada sebuah tabel laman web.

Daftar Pustaka

1. Bellinger, G., Castro, D. and Mills, A., 2004. Data, information, knowledge, and wisdom.
2. John Walker, S., 2014. Big data: A revolution that will transform how we live, work, and think.
3. Schmidt, E. and Cohen, J., 2014. The new digital age: Transforming nations, businesses, and our lives. Vintage.
4. Neff, D.J. and Moss, R.C., 2011. The future of nonprofits: Innovate and thrive in the digital age. John Wiley & Sons.
5. Brynjolfsson, E. and McAfee, A., 2014. The second machine age: Work, progress, and prosperity in a time of brilliant technologies. WW Norton & Company.
6. Li, S., Zhang, H., Jia, Z., Li, Z., Zhang, C., Li, J., … Shan, Z. (2019). A dataflow-driven approach to identifying microservices from monolithic applications. Journal of Systems and Software, 157, 110380. https://doi.org/10.1016/j.jss.2019.07.008
7. Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software, 150, 77–97. https://doi.org/10.1016/j.jss.2019.01.001
8. Wan, X., Guan, X., Wang, T., Bai, G., & Choi, B.-Y. (2018). Application deployment using Microservice and Docker containers: Framework and optimization. *Journal of Network and Computer Applications, 119*, 97–109. https://doi.org/10.1016/j.jnca.2018.07.003
9. Kautsar, I. A., Kubota, S.-I., Musashi, Y., & Sugitani, K. (2014, December). Redefining Data Provider: The REST

approach To solve Indonesia Lecturer administrative problems. 175–178. https://doi.org/10.1109/TALE.2014.7062614
10. Kautsar, I. A., & Sarno, R. (2019). A Supportive Tool for Project Based Learning and Laboratory Based Education. International Journal on Advanced Science, Engineering and Information Technology, 9(2), 630–639.
11. Jones, K.J., Bejtlich, R. and Rose, C.W., 2006. Real digital forensics: computer security and incident response (pp. 3-4). Addison-Wesley.
12. Souders, S., 2009. Even faster web sites: performance best practices for web developers. " O'Reilly Media, Inc.".

Biodata Penulis

Irwan A. Kautsar. Kelahiran tahun 1982, menyelesaikan Studi S1 di Teknik Informatika ITS Surabaya (2004-2008); S2 Teknik Informatika ITS Surabaya (2009-2012) dan S3 di Kumamoto University, Kumamoto, Jepang (2012-2016). Saat ini aktif mengajar di Program Studi Informatika Fakultas Sains dan Teknologi Universitas Muhammadiyah Sidoarjo.

Bidang penelitian penulis ialah Network Security, Open System, Sistem Terdistribusi Konten Pembelajaran (Distributed Learning

Object), Desentralisasi Jaringan Komputer dan Web Services.

Publikasi yang dilakukan penulis dapat dilihat pada laman berikut: http://hepidad.github.io/pubs.

Penulis dapat dijangkau melalui surel irwan@umsida.ac.id.

Teknologi Informasi yang dikembangkan, sebagian besar dalam bentuk sebuah software/aplikasi. Aplikasi Berbasis Teknologi Web merupakan jawaban integrasi data pada aplikasi multi platform dengan menggunakan layanan web (web service). Buku ini merupakan buku ajar pendamping mata kuliah pengembangan aplikasi web. Dimana dalam buku ajar ini mahasiswa dikenalkan dengan pemanfaatan microframework dalam mengembangkan web service dan microservices.

Irwan A. Kautsar, saat ini aktif mengajar di Prodi Informatika Fakultas Sains dan Teknologi Universitas Muhammadiyah Sidoarjo. Bidang penelitian penulis ialah Network Security, Open System, Sistem Terdistribusi dan Konten Pembelajaran (Distributed Learning Object), Desentralisasi Jaringan Komputer dan Web Services. Publikasi yang dilakukan penulis dapat dilihat pada laman berikut: http://hepidad.github.io/ pubs.

Cover Design: Stay@HomeMom

ISBN 978-623-7578-53-6