# BAB 4 IMPLEMENTASI

Pada bab ini akan dijelaskan proses implementasi yang dilakukan untuk membangun *web service* untuk program *Morphological Analyzer* yang telah dijelaskan sebelumnya, yang diberi nama "MorphoWSService", serta implementasi *web application* pada *client side* untuk melalukan uji coba pada *web service* yang dihasilkan. Seperti yang telah dijelaskan pada Bab 3, terdapat tiga hal yang harus diimplementasikan, yaitu *web application* pada *server side*, *file* WSDL yang mendeskripsikan *web service* ini, dan *web application* pada *client side*. Proses implementasi ketiga hal tersebut akan dijelaskan pada Subbab 4.1 hingga Subbab 4.3.

## 1.1 Implementasi *Web Application* pada *Server Side*

Pada subbab ini akan dijelaskan mengenai proses implementasi *web application* pada *server side* yang akan berkomunikasi dengan program *Morphological Analyzer*. Implementasi *web application* pada *server side* ini menggunakan bahasa pemrograman Java dengan IDE Netbeans 6.1 dan *web server* Apache Tomcat 6.0.

Seperti perancangan yang telah dijelaskan pada Bab 3, terdapat lima buah fungsi atau *method* yang akan diimplementasikan dalam aplikasi ini, yaitu fungsi **`applyUp`**, **`applyDown`**, **`applyUpAll`**, **`applyDownAll`**, dan **`doAnalyze`**. *Pseudocode* dari kelima fungsi tersebut dapat dilihat pada Gambar 0.1.

```
function applyUp( str )
    return apply( "up", false, str )
end function

function applyDown( str )
    return apply( "down", false, str )
end function

function applyUpAll( str )
    return apply( "up", true, str )
end function

function applyDownAll( str )
```

```
    return apply( "down", true, str )
end function

function doAnalyze( language, text )
    instantiate collection (result)
    split text by " " to array (word)
    i := 0
    while i < length of word
        analyse := apply( "up", false, word[i] )
        create morpheme based on analyse
        add morpheme to result
        increment i
    end while
    return result
end function
```

Gambar 0.1 *Pseudocode* fungsi **applyUp**, **applyDown**, **applyUpAll**, **applyDownAll**, dan **doAnalyze**

Proses komunikasi dengan program *Morphological Analyzer* yang dilakukan di dalam keempat fungsi tersebut tidak berbeda jauh. Perbedaan yang ada hanyalah pada jenis perintah dan spesifikasi masukan dan keluaran yang berbeda. Oleh karena itu, untuk menghindari redundansi pada *code* yang dibuat, dibuatlah sebuah fungsi yang lebih umum, yaitu fungsi **apply**, yang akan dipanggil oleh keempat fungsi tersebut. *Pseudocode* dari fungsi **apply** dapat dilihat pada Gambar 0.2.

```
function apply( command, all, text )
    result := ""
    run command prompt process
    run program Morphological Analyzer
    load binary file myfile.fst
    input command and text
    read the output
    skip unnecessary output

    if all = true
        line := get one line of output
        while line != "bye."
            result := line + " "
            line := get one line of output
        end while
    else
        line := get one line of output
        if line != "bye."
            result := line + " "
        end if
    end if
```

```
    return result
end function
```

Gambar 0.2 *Pseudocode* fungsi **apply**

Untuk mendukung proses komunikasi antara aplikasi ini dengan program *Morphological Analyzer*, dibuatlah sebuah proses untuk menjalankan program *command prompt*. Proses ini akan dibuat dengan menggunakan obyek **Process** dan **ProcessBuilder** pada Java. Melalui program *command prompt* ini, program *Morphological Analyzer* akan dipanggil, diberikan masukan, dan diambil hasil keluarannya. Proses pemberian masukan atau perintah pada program *command prompt* menggunakan obyek **InputStreamReader** dan obyek **BufferedWriter**. Sedangkan, proses pengambilan keluaran dari program *command prompt* menggunakan obyek **OutputStreamWriter** dan obyek **BufferedReader**.

Setelah program *Morphological Analyzer* dijalankan melalui program *command prompt*, hal berikutnya yang dilakukan adalah membuka *binary file* **myfile.fst** yang berisi hasil kompilasi dari *rule-rule* yang ada pada program *Morphological Analyzer* tersebut. Pembuatan *binary file* ini adalah sebuah usaha untuk menghindari proses kompilasi yang cukup lama. Perbedaan waktu eksekusi dengan menggunakan *binary file* ini dengan waktu eksekusi tanpa menggunakan *binary file* ini, atau dengan melakukan kompilasi terlebih dahulu, cukup signifikan. Sebagai perbandingan, waktu eksekusi untuk sebuah masukan tanpa menggunakan *binary file* ini dapat mencapai 710 detik, sedangkan dengan menggunakan *binary file* ini waktu eksekusi tersebut dapat dikurangi hingga mencapai kurang dari 5 detik, atau lebih dari seratus kali lebih cepat[1].

Proses pembuatan *binary file* ini adalah dengan melakukan kompilasi dan memasukkan hasilnya ke dalam *binary file* **myfile.fst** dengan menggunakan perintah "**save stack myfile.fst**" (Beesley & Karttunen, 2003). Setelah dimasukkan ke dalam *binary file* **myfile.fst**, hasil kompilasi dapat diakses dengan menggunakan perintah "**load stack myfile.fst**".

[1] Eksekusi dilakukan pada PC dengan prosesor Intel® Celeron® M 1.50 GHz, 752 MB of RAM. Eksekusi yang dilakukan adalah pemanggilan *method* **applyUp("memukul")**.

Setelah membuka *binary file* hasil kompilasi, masukan diberikan kepada program *Morphological Analyzer* oleh obyek **`BufferedWriter`** sesuai dengan perintah yang diberikan. Terdapat dua jenis perintah yang dapat diberikan, yaitu perintah "apply up" dan perintah "apply down". Setelah semua masukan telah diberikan, hasil keluaran dari program *Morphological Analyzer* akan diambil oleh obyek **`BufferedReader`**. Contoh keluaran dari program tersebut dapat dilihat pada Gambar 0.3.

```
xerox-windows

D:\xerox-windows>xfst.exe
Opening
'myfile.fst'
Closing
'myfile.fst'
ma+Verb
mak+Noun
makan+BareNoun
makan+BareVerb+UV
```

Gambar 0.3 Contoh keluaran program *Morphological Analyzer*

Pada contoh keluaran di atas, bagian keluaran yang penting terdapat pada empat baris terakhir. Hasil keluaran pada baris-baris sebelumnya tidaklah penting untuk dijadikan keluaran fungsi **`apply`**, sehingga baris-baris tersebut tidak perlu dibaca oleh **`BufferedReader`**.

Selanjutnya, dilakukan proses manipulasi hasil keluaran sesuai dengan perintah yang diberikan. Jika fungsi yang dipanggil adalah fungsi **`applyUpAll`** atau fungsi **`applyDownAll`**, berarti semua hasil keluaran harus ditampilkan dan dimasukkan ke dalam variabel **`result`** dengan ' ' (spasi) sebagai karakter pembatas dari dua buah keluaran. Sedangkan, jika fungsi yang dipanggil adalah fungsi **`applyUp`** atau fungsi **`applyDown`**, hasil keluaran yang dimasukkan ke dalam variabel **`result`** hanyalah hasil keluaran yang pertama saja. Terakhir, isi variabel **`result`** akan dikembalikan sebagai keluaran dari fungsi **`apply`**.

### 1.2 Pembuatan *File* WSDL dan *File* XML-Schema

Setelah melakukan implementasi *web application* pada *server side*, dilakukan pembuatan *file* WSDL yang akan mendeskripsikan operasi yang disediakan oleh *web service* "MorphoWSService" serta lokasi dari *web service* tersebut. *File* tersebut dihasilkan oleh *tools* Netbeans 6.1 dengan melakukan proses "Undeploy and Deploy". Isi *file* WSDL yang dihasilkan, yaitu *file* "MorphoWSService.wsdl", dapat dilihat pada Lampiran B. Selain itu, juga dihasilkan sebuah *file* XML Schema, yaitu *file* "MorphoWSService_schema1.xsd", yang berisi penjelasan dari tipe data yang digunakan oleh *web service* yang menjadi referensi dari *file* WSDL. Isi dari *file* XML Schema tersebut dapat dilihat pada Lampiran C.

Terdapat lima buah elemen utama yang menjadi isi dari *file* WSDL, yaitu elemen **<types>**, **<message>**, **<portType>**, **<binding>**, dan **<service>**. Kelima elemen tersebut akan dijelaskan pada Subbab 4.2.1 hingga Subbab 4.2.5.

#### 1.2.1 Penjelasan Elemen `<types>`

Elemen **<types>** digunakan untuk mendefinisikan tipe data yang digunakan oleh *web service*. Penjabaran dari elemen **<types>** yang ada pada *file* "MorphoWSService.wsdl" dapat dilihat pada Gambar 0.4. Elemen **<types>** pada Gambar 0.4 menjelaskan bahwa tipe data yang digunakan oleh *web service* "MorphoWSService" akan didefinisikan pada *file* XML Schema yang bernama "MorphoWSService_schema1.xsd". Penggunaan *namespace* "http://morpho.me.org/" di sini hanyalah untuk keperluan percobaan saja. Pada saat *web service* ini akan di-*deploy* pada Language Grid, *namespace* ini akan disesuaikan dengan *namespace* yang digunakan oleh Language Grid.

```
<types>
  <xsd:schema>
    <xsd:import namespace="http://morpho.me.org/"
      schemaLocation="MorphoWSService_schema1.xsd"/>
  </xsd:schema>
</types>
```

Gambar 0.4 Penjabaran elemen **<types>** pada *file* "MorphoWSService.wsdl"

### 1.2.2 Penjelasan Elemen `<message>`

Elemen **<message>** digunakan untuk mendefinisikan *message* yang digunakan oleh *web service*. Penjabaran dari elemen **<message>** yang ada pada *file* "MorphoWSService.wsdl" dapat dilihat pada Gambar 0.5.

```
<message name="applyUp">
  <part name="parameters" element="tns:applyUp"/>
</message>

<message name="applyUpResponse">
  <part name="parameters" element="tns:applyUpResponse"/>
</message>

<message name="applyDown">
  <part name="parameters" element="tns:applyDown"/>
</message>

<message name="applyDownResponse">
  <part name="parameters" element="tns:applyDownResponse"/>
</message>

<message name="applyUpAll">
  <part name="parameters" element="tns:applyUpAll"/>
</message>

<message name="applyUpAllResponse">
  <part name="parameters" element="tns:applyUpAllResponse"/>
</message>

<message name="applyDownAll">
  <part name="parameters" element="tns:applyDownAll"/>
</message>

<message name="applyDownAllResponse">
  <part name="parameters" element="tns:applyDownAllResponse"/>
</message>

<message name="doAnalyze">
  <part name="parameters" element="tns:doAnalyze"/>
</message>

<message name="doAnalyzeResponse">
  <part name="parameters" element="tns:doAnalyzeResponse"/>
</message>
```

Gambar 0.5 Penjabaran elemen **<messages>** pada *file* "MorphoWSService.wsdl"

Terdapat sepuluh buah *message* yang digunakan pada *web service* "MorphoWSService", yaitu *message* "applyUp", "applyDown", "applyUpAll", "applyDownAll", "doAnalyze", "applyUpResponse", "applyDownResponse", "applyUpAllResponse", "applyDownAllResponse", dan "doAnalyzeResponse". Di dalam masing-masing elemen **<message>**, terdapat elemen **<part>** yang menjelaskan parameter yang menyertai *message* tersebut. Tipe data dari parameter tersebut diberikan pada *file* "MorphoWSService_schema1.xsd".

### 1.2.3 Penjelasan Elemen <portType>

Elemen **<portType>** digunakan untuk mendefinisikan operasi yang disediakan oleh *web service*. Penjabaran dari elemen **<portType>** yang ada pada *file* "MorphoWSService.wsdl" dapat dilihat pada Gambar 0.6.

```
<portType name="MorphoWS">
  <operation name="applyUp">
    <input message="tns:applyUp"/>
    <output message="tns:applyUpResponse"/>
  </operation>

  <operation name="applyDown">
    <input message="tns:applyDown"/>
    <output message="tns:applyDownResponse"/>
  </operation>

  <operation name="applyUpAll">
    <input message="tns:applyUpAll"/>
    <output message="tns:applyUpAllResponse"/>
  </operation>

  <operation name="applyDownAll">
    <input message="tns:applyDownAll"/>
    <output message="tns:applyDownAllResponse"/>
  </operation>

  <operation name="doAnalyze">
    <input message="tns:doAnalyze"/>
    <output message="tns:doAnalyzeResponse"/>
  </operation>
</portType>
```

Gambar 0.6 Penjabaran elemen **<portType>** pada *file* "MorphoWSService.wsdl"

Terdapat lima buah operasi yang disediakan oleh *web service* "MorphoWSService", yaitu operasi "applyUp", "applyDown", "applyUpAll", "applyDownAll", dan "doAnalyze". Masing-masing operasi tersebut akan menjalankan *method* yang bersesuaian pada *web application* pada *server side*, dan memiliki masukan dan keluaran berupa *message* yang telah didefinisikan sebelumnya, dan telah dijelaskan pada Subbab 4.2.2.

### 1.2.4 Penjelasan Elemen `<binding>`

Elemen **`<binding>`** digunakan untuk mendefinisikan protokol komunikasi yang digunakan oleh *web service*. Penjabaran dari elemen **`<binding>`** yang ada pada *file* "MorphoWSService.wsdl" dapat dilihat pada Gambar 0.7.

```
<binding name="MorphoWSPortBinding" type="tns:MorphoWS">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>

  <operation name="applyUp">
    <soap:operation soapAction=""/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
  </operation>

  <operation name="applyDown">
    <soap:operation soapAction=""/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
  </operation>

  <operation name="applyUpAll">
    <soap:operation soapAction=""/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
  </operation>

  <operation name="applyDownAll">
    <soap:operation soapAction=""/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
  </operation>

  <operation name="doAnalyze">
    <soap:operation soapAction=""/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
```

```
  </operation>
</binding>
```

Gambar 0.7 Penjabaran elemen **<binding>** pada *file* "MorphoWSService.wsdl"

*Web service* "MorphoWSService" menggunakan SOAP dengan HTTP sebagai protokol pengiriman dan pertukaran *message*. Informasi ini diberikan pada atribut **transport** pada elemen **<soap:binding>**. Selanjutnya, didefinisikan bagaimana *message* yang menjadi masukan dan keluaran dari setiap operasi dikodekan (*encode*). Pada *web service* ini, digunakan *encoding literal* untuk setiap *message* masukan dan keluaran dari setiap operasi. Nama operasi yang didefinisikan protokolnya pada elemen ini disesuaikan dengan nama operasi yang telah didefinisikan sebelumnya pada elemen **<portType>**, yaitu operasi "applyUp", "applyDown", "applyUpAll", "applyDownAll", dan "doAnalyze".

### 1.2.5 Penjelasan Elemen `<service>`

Elemen **<service>** digunakan untuk memberikan informasi lokasi *web service*. Pada kasus ini, *web service* "MorphoWSService" dapat diakses melalui lokasi http://localhost:8080/MorphoWSService/MorphoWS. Penjabaran dari elemen **<service>** yang ada pada *file* "MorphoWSService.wsdl" dapat dilihat pada Gambar 0.8.

```
<service name="MorphoWSService">
  <port name="MorphoWSPort" binding="tns:MorphoWSPortBinding">
    <soap:address
      location="http://localhost:8080/MorphoWSService/MorphoWS"/>
  </port>
</service>
```

Gambar 0.8 Penjabaran elemen **<service>** pada *file* "MorphoWSService.wsdl"

## 1.3 Implementasi *Web Application* pada *Client Side*

Untuk membangun *web application* pada *client side*, digunakan bahasa pemrograman PHP. Alasan pemilihan bahasa pemrograman PHP ini salah satunya adalah untuk menunjukkan sifat *multiplatform* yang dimiliki oleh *web service*. Dalam

kasus ini, *web application* pada *client side* yang menggunakan *platform* PHP tetap dapat menggunakan *web service* yang *web application* pada *server side*-nya menggunakan *platform* yang berbeda, yaitu Java.

Digunakan *library* NuSOAP untuk mengimplementasikan aplikasi *web* berbasis PHP pada *client side*. NuSOAP adalah kumpulan kelas-kelas PHP yang dapat digunakan untuk membuat atau membangun SOAP *web services* (Scott Nichol, 2004).

*Web application* pada *client side* ini diimplementasikan dalam dua buah *file*, yaitu *file* client.php dan *file* AjaxClient.php. Isi dari kedua *file* tersebut dapat dilihat pada Lampiran D.

Dilakukan sedikit perubahan pada *library* NuSOAP, yaitu pergantian nama fungsi **`soapclient`** menjadi **`soapclientw`**. Perubahan ini dilakukan untuk menghindari konflik nama fungsi **`soapclient`** yang telah ada pada PHP secara umum. Dengan menggunakan fungsi ini, dibuatlah sebuah *client instance* dengan membuat *code* seperti yang ada pada Gambar 0.9.

```
require_once('libs/nusoap.php');
$wsdl = "http://localhost:8080/MorphoWSApplication/MorphoWS?wsdl";
$client=new soapclientw($wsdl);
```

Gambar 0.9 Pembuatan client instance menggunakan *library* NuSOAP

Selanjutnya, dengan menggunakan *client instance* ini, fungsi-fungsi yang disediakan oleh *web service* dapat diakses seperti *code* yang ditunjukkan pada Gambar 0.10. Pada gambar tersebut, *client instance* memanggil fungsi **`applyUp`** dengan parameter **`arg0`** berisi isi variabel **`$inputString`**. Hasil keluaran dari pemanggilan fungsi tersebut dimasukkan ke dalam variabel **`$outputString`**.

```
$param=array('arg0'=>$inputString);
$outputString = $client->call('applyUp', $param, $ns );
```

Gambar 0.10 Pemanggilan fungsi **`applyUp`** dengan menggunakan *library* NuSOAP

Secara garis besar, hal yang dilakukan *web application* pada *client side* adalah meneruskan *request* dari *client* untuk menggunakan *web service*

"MorphoWSService". *Request* yang diberikan *client* berupa operasi yang ingin digunakan dan *text* yang ingin diproses. Terdapat lima buah operasi yang dapat digunakan, yaitu operasi "Apply Up", "Apply Down", "Apply Up All", "Apply Down All", dan "Do Analyze". *Text* yang dimasukkan *client* akan diubah menjadi *lower case* dan setiap karakter yang bukan merupakan huruf akan diabaikan. *Screenshot* antarmuka dari *web application* pada *client side* ini dapat dilihat pada Gambar 0.11.


Gambar 0.11 *Screenshot web application* pada *client side*

Pemanggilan *method* yang bersesuaian dengan operasi tersebut akan dilakukan ketika pengguna menekan tombol "Process". Selanjutnya, hasil keluaran beserta dengan waktu eksekusi akan ditampilkan pada bagian "Results" dan "Time". Waktu eksekusi yang dihitung adalah waktu AjaxClient.php, yang meminta *request* pada *web service*, dijalankan. Waktu ekesekusi ini didapatkan dengan menghitung

selisih waktu awal dan waktu akhir AjaxClient.php dijalankan. Waktu awal dan waktu akhir, dalam hitungan mikrodetik, didapatkan dengan memanggil fungsi `microtime(true)` pada PHP.

**UNIVERSITAS INDONESIA**

# BAB 5 HASIL IMPLEMENTASI DAN UJI COBA

Pada bab ini akan dipaparkan hasil dari implementasi yang telah dijelaskan pada Bab 4. Selain itu, akan dijelaskan pula uji coba yang telah dilakukan beserta dengan hasil uji coba tersebut.

Uji coba dari *web service Morphological Analyzer* dilakukan dengan memberikan *request* melalui *web application* pada *client side* yang telah dibangun[1]. *Request* diberikan dengan menjalankan operasi "Apply Up", "Apply Down", "Apply Up All", dan "Apply Down All", dan "Do Analyze" yang telah dibangun pada *web application* pada *client side*. Kemudian, hasil keluaran yang diperoleh akan dibandingkan dengan hasil keluaran yang diharapkan. Hasil keluaran yang diharapkan adalah hasil keluaran pada program *Morphological Analyzer*, dengan penyesuaian jumlah hasil keluaran berdasarkan operasi yang dilakukan. Operasi "Apply Up" dan "Apply Down" hanya akan mengeluarkan satu buah hasil keluaran saja, yaitu hasil keluaran pertama pada program *Morphological Analyzer*, sedangkan operasi "Apply Up All" dan "Apply Down All" akan mengeluarkan semua hasil keluaran yang diberikan oleh program *Morphological Analyzer*.

Secara umum, uji coba hanya perlu dilakukan untuk masukan-masukan tertentu saja yang cukup representatif dari segi jumlah keluaran yang dihasilkan pada program *Morphological Analyzer*.

## 1.1 Uji Coba Operasi "Apply Up"

Operasi "Apply Up" akan memanggil *method* `applyUp` pada *web application* pada *server side*. Uji coba dilakukan pada empat buah masukan yang cukup representatif, yaitu *string* masukan "aeiou" yang menghasilkan keluaran berupa *string* kosong pada *Morphological Analyzer*[2], *string* masukan "makan" yang menghasilkan empat buah keluaran, *string* "membaca" yang menghasilkan sebuah keluaran, dan

[1] Uji coba dilakukan pada PC dengan prosesor Intel® Celeron® M 1.50 GHz, 752 MB of RAM. *Web server* yang digunakan adalah Apache Tomcat 6.0.

[2] Kata "aeiou" bukanlah kata dasar atau kata berimbuhan dalam Bahasa Indonesia, sehingga akan menghasilkan keluaran berupa *string* kosong pada program *Morphological Analyzer*.

*string* "berlari-larian" yang menghasilkan sebuah keluaran juga. Keluaran yang diharapkan adalah keluaran pertama (baris teratas) dari keluaran pada program *Morphological Analyzer*. Hasil uji coba yang dilakukan pada operasi "Apply Up" dapat dilihat pada Tabel 0.1[3].

Tabel 0.1 Hasil uji coba operasi "Apply Up"

| No | Masukan | Keluaran pada *Morphological Analyzer* | Keluaran pada *client side* | Hasil |
|---|---|---|---|---|
| 1 | aeiou | <string kosong> | <string kosong> | Benar |
| 2 | makan | ma+Verb<br>mak+Noun<br>makan+BareNoun<br>makan+BareVerb+UV | ma+Verb | Benar |
| 3 | membaca | baca+Verb+AV | baca+Verb+AV | Benar |
| 4 | berlari-larian | lari+Verb+Redup | lari+Verb+Redup | Benar |

Dari keempat masukan yang dicobakan, hasil keluaran pada *client side* sesuai dengan hasil keluaran yang diharapkan, yaitu hasil keluaran pertama pada program *Morphological Analyzer*. *Screenshot* untuk uji coba nomor 1 dapat dilihat pada Gambar 0.1.

### 1.2 Uji Coba Operasi "Apply Down"

Operasi "Apply Down" akan memanggil *method* `applyDown` pada *web application* pada *server side*. Uji coba dilakukan pada tiga buah masukan yang cukup representatif, yaitu *string* masukan "pukul" yang menghasilkan keluaran berupa *string* kosong pada *Morphological Analyzer*[4], *string* masukan "pukul+Verb+AV" yang menghasilkan sembilan buah keluaran, dan *string* "pukul+BareNoun" yang menghasilkan sebuah keluaran.

[3] Hasil keluaran pada *web application* pada *client side* sama dengan hasil keluaran pada program *Morphological Analyzer*. Namun, program *Morphological Analyzer* yang digunakan masih belum sempurna. Keterbatasan yang dimiliki program ini dapat dilihat pada (Pisceldo, 2008).

[4] Tidak terdapat *tag* yang memberikan informasi sintaksis dan semantik dalam *string* masukan, sehingga akan menghasilkan keluaran berupa *string* kosong pada program *Morphological Analyzer*.


Gambar 0.1 *Screenshot* uji coba pertama pada operasi "Apply Up"

Keluaran yang diharapkan adalah keluaran pertama (baris teratas) dari keluaran pada program *Morphological Analyzer*. Hasil uji coba yang dilakukan pada operasi “Apply Down” dapat dilihat pada Tabel 0.2.

Tabel 0.2 Hasil uji coba operasi "Apply Down"

| No | Masukan | Keluaran pada *Morphological Analyzer* | Keluaran pada *client side* | Hasil |
|---|---|---|---|---|
| 1 | pukul | <string kosong> | <string kosong> | Benar |
| 2 | pukul+Verb+AV | mengepukulkan<br>menerpukulkan<br>memperpukul<br>memperpukulkan<br>memperpukuli<br>memberpukulkan<br>memukul<br>memukulkan<br>memukuli | mengepukulkan | Benar |
| 3 | pukul+BareNoun | pukul | pukul | Benar |

Dari ketiga masukan yang dicobakan, hasil keluaran pada *client side* sesuai dengan hasil keluaran yang diharapkan, yaitu hasil keluaran pertama pada program *Morphological Analyzer*. *Screenshot* untuk uji coba nomor 2 dapat dilihat pada Gambar 0.2.


Gambar 0.2 *Screenshot* uji coba kedua pada operasi "Apply Down"

## 1.3 Uji Coba Operasi "Apply Up All"

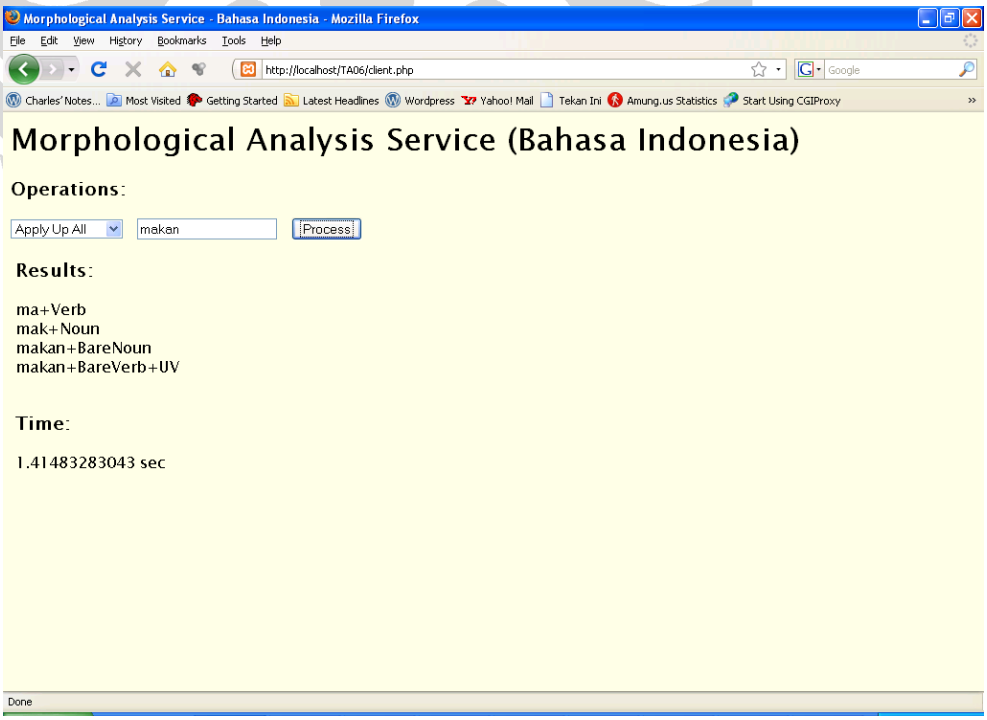
Operasi "Apply Up All" akan memanggil *method* `applyUpAll` pada *web application* pada *server side*. Uji coba dilakukan pada tiga buah masukan yang cukup representatif, yaitu *string* masukan "aeiou" yang menghasilkan keluaran berupa *string* kosong pada *Morphological Analyzer*, *string* masukan "makan" yang menghasilkan empat buah keluaran, dan *string* "membaca" yang menghasilkan sebuah keluaran. Keluaran yang diharapkan adalah semua hasil keluaran pada program *Morphological*

*Analyzer*. Hasil uji coba yang dilakukan pada operasi "Apply Up All" dapat dilihat pada Tabel 0.3.

Tabel 0.3 Hasil uji coba operasi "Apply Up All"

| No | Masukan | Keluaran pada *Morphological Analyzer* | Keluaran pada *client side* | Hasil |
|---|---|---|---|---|
| 1 | aeiou | <string kosong> | <string kosong> | Benar |
| 2 | makan | ma+Verb<br>mak+Noun<br>makan+BareNoun<br>makan+BareVerb+UV | ma+Verb<br>mak+Noun<br>makan+BareNoun<br>makan+BareVerb+UV | Benar |
| 3 | membaca | baca+Verb+AV | baca+Verb+AV | Benar |

Dari ketiga masukan yang dicobakan, hasil keluaran pada *client side* sesuai dengan hasil keluaran yang diharapkan, yaitu semua hasil keluaran pada program *Morphological Analyzer*. *Screenshot* untuk uji coba nomor 2 dapat dilihat pada Gambar 0.3.




Gambar 0.3 *Screenshot* uji coba kedua pada operasi "Apply Up All"

### 1.4 Uji Coba Operasi "Apply Down All"

Operasi "Apply Down All" akan memanggil *method* `applyDownAll` pada *web application* pada *server side*. Uji coba dilakukan pada tiga buah masukan yang cukup representatif, yaitu *string* masukan "pukul" yang menghasilkan keluaran berupa *string* kosong pada *Morphological Analyzer*, *string* masukan "pukul+Verb+AV" yang menghasilkan sembilan buah keluaran, dan *string* "pukul+BareNoun" yang menghasilkan sebuah keluaran. Keluaran yang diharapkan adalah semua hasil keluaran pada program *Morphological Analyzer*. Hasil uji coba yang dilakukan pada operasi "Apply Down All" dapat dilihat pada Tabel 0.4.

Tabel 0.4 Hasil uji coba operasi "Apply Down All"

| No | Masukan | Keluaran pada *Morphological Analyzer* | Keluaran pada *client side* | Hasil |
|---|---|---|---|---|
| 1 | pukul | <string kosong> | <string kosong> | Benar |
| 2 | pukul+Verb+AV | mengepukulkan<br>menerpukulkan<br>memperpukul<br>memperpukulkan<br>memperpukuli<br>memberpukulkan<br>memukul<br>memukulkan<br>memukuli | mengepukulkan<br>menerpukulkan<br>memperpukul<br>memperpukulkan<br>memperpukuli<br>memberpukulkan<br>memukul<br>memukulkan<br>memukuli | Benar |
| 3 | pukul+BareNoun | pukul | pukul | Benar |

Dari ketiga masukan yang dicobakan, hasil keluaran pada *client side* sesuai dengan hasil keluaran yang diharapkan, yaitu semua hasil keluaran pada program *Morphological Analyzer*. *Screenshot* untuk uji coba nomor 2 dapat dilihat pada Gambar 0.4.

### 1.5 Uji Coba Operasi "Do Analyze"

Operasi "Do Analyze" akan memanggil *method* `doAnalyze` pada *web application* pada *server side*. Uji coba dilakukan pada empat buah masukan yang

dapat dilihat pada Tabel 0.5, dan telah dicocokkan sesuai dengan hasil keluaran pada *Morphological Analyzer*.


Gambar 0.4 *Screenshot* uji coba kedua pada operasi "Apply Down All"

Tabel 0.5 Hasil uji coba operasi "Do Analyze"

| No | Masukan | Keluaran pada *client side* | | | Hasil |
|---|---|---|---|---|---|
| | | word | original | part of speech | |
| 1 | aeiou | aeiou | | unknown | Benar |
| 2 | makan | makan | ma | verb | Benar |
| 3 | makan bubur | makan | ma | verb | Benar |
| | | bubur | bubur | noun | |
| 4 | Pengantar pada awal buku berisi tujuan penulisan buku teks pelajaran TIK, sistematika, dan cara-cara pengajaran termasuk materi yang diberikan. | pengantar | antar | noun | Benar |
| | | pada | pada | noun | |
| | | awal | awal | noun | |
| | | buku | buku | noun | |
| | | berisi | isi | verb | |
| | | tujuan | tuju | noun | |
| | | penulisan | tulis | noun | |
| | | buku | buku | noun | |
| | | teks | teks | adjective | |
| | | pelajaran | | unknown | |
| | | tik | tik | noun | |
| | | sistematika | sistematika | noun | |
| | | dan | dan | noun | |

| | | | | | |
|---|---|---|---|---|---|
| | | cara-cara | cara | noun | |
| | | pengajaran | ajar | noun | |
| | | termasuk | masuk | verb | |
| | | materi | materi | adjective | |
| | | yang | yang | other | |
| | | diberikan | beri | verb | |

Dari keempat masukan yang dicobakan, hasil keluaran pada *client side* sesuai dengan hasil keluaran yang diharapkan. *Screenshot* untuk uji coba nomor 4 dapat dilihat pada Gambar 0.5.


Gambar 0.5 *Screenshot* uji coba keempat pada operasi "Do Analyze"

**UNIVERSITAS INDONESIA**