Bab II
DASAR TEORI

2.1. *Mobile Learning*

2.1.1. Konsep Umum *Mobile Learning*

Pengertian dari *Mobile Learning* mengalami perubahan seiiring dengan munculnya teknologi-teknologi baru. Dalam penelitiannya, Susan Smith Nash [5] mengungkapkan bahwa pengertian *mobile learning* adalah istilah untuk materi pembelajaran maupun aktifitas belajar yang disampaikan dengan menggunakan media perangkat bergerak yang mengakomodasi keterbatasan penyampaian multimedia, terutama dalam bentuk suara, gambar, animasi dan teks (Susan, 2007, p1).

Sementara Kineo dalam *review*-nya, mengungkapkan bahwa pengertian *mobile learning* adalah kemampuan belajar yang tidak terikat kepada tempat maupun waktu, yang difasilitasi oleh perangkat bergerak [6]. Pengertian *mobile learning* yang digunakan dalam tulisan ini adalah materi pembelajaran yang disampaikan dengan menggunakan media perangkat bererak yang tidak terikat kepada tempat maupun waktu. Karakteristik dari *mobile learning* yang perlu diperhatikan adalah [6]:

1. *Ubiquitous*, yang artinya materi *mobile learning* bisa diakses di mana saja, berkaitan dengan lokasi belajar.
2. *Bite sized*, yang artinya ukuran dari materi *mobile learning* yang diakses harus disampaikan dalam durasi yang singkat. Hal ini untuk mengantisipasi siswa yang mengakses *mobile learning* pada situasi yang penuh interupsi yang dapat mengganggu konsentrasi siswa.
3. *On demand*, yang artinya *mobile learning* harus sanggup menyampaikan materi dikala dibutuhkan oleh siswa. Memaksimalkan penyampaian materi pembelajaran saat dibutuhkan oleh siswa.
4. *Typically blended*, yang artinya *mobile learning* sudah biasa digunakan bersama metode pembelajaran yang lain. *Mobile Learning* cenderung digunakan sebagai alat bantu untuk meningkatkan pemahaman siswa selain yang telah disampaikan dengan metode lain, misal belajar di kelas.

5. *Collaborative*, yang artinya *mobile learning* harus dapat memanfaatkan kemampuan perangkatnya yang memiliki kemampuan komunikasi. Dengan kemampuan komunikasi ini, sekelompok orang dapat belajar secara bersama dan saling berbagi pengetahuan satu sama lain. Dengan demikian, *mobile learning* memiliki potensi untuk membuat komunitas *mobile*, atau paling tidak, interaksi dengan guru dapat dilakukan melalui *mobile learning*.
6. *Location dependent*, yang artinya perangkat bergerak memiliki potensi untuk menyampaikan materi yang sesuai dengan posisi siswa. Misalnya menyampaikan tips penjualan bagi sales yang sesuai dengan pelanggan yang akan dikunjungi terkait dengan tempat pertemuan. Sadar akan lokasi bisa didukung oleh berbagai teknologi termasuk prinsip segitiga dari jaringan seluler atau GPS (Global Positioning system), yang akhirnya dapat mengirimkan materi khusus sesuai dengan lokasi siswa.

Perangkat elektronik yang disebut sebagai perangkat bergerak, dibagi menjadi beberapa kategori, yaitu PDA atau *smartphone*, Telepon Digital, dan perangkat *non-telephony* termasuk MP3 *players* [6]. Tipe utama dari masing-masing kategori adalah seperti berikut :

1. PDA atau *Smartphone*

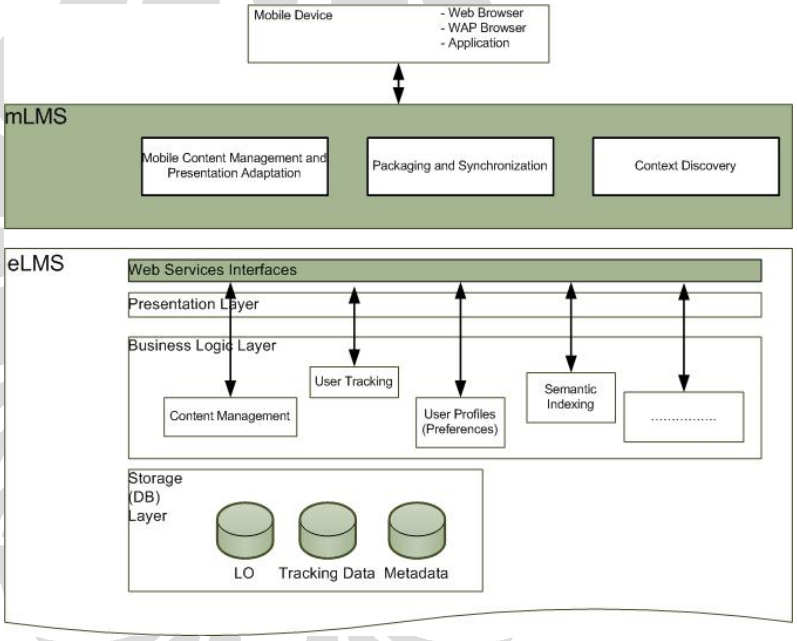
   Ada tiga bentuk utama dari teknologi PDA yaitu Palm *Operating System*, Pocket PC dari Microsoft ® dan *Smartphone*. *Smartphone* biasa didefinisikan sebagai sebuah perangkat dengan layar lebar, berpusat pada data yang didesign untuk menawarkan fungsi telepon bersama fungsi PDA. Kategori ini juga memasukkan *Blackberry* dari Research in Motion dan model Motorola.
2. *Digital Phone*

   Termasuk dalam kategori ini adalah berbagai perangkat yang telah mendukung teknologi 3G dengan layar lebar dan kemampuan suara/video/flash/java. Termasuk dalam kategori ini misalnya adalah *handset* yang dibuat oleh Nokia, Sony Ericsson dan juga *Blackberry* yang dibuat oleh Research in Motion.
3. Perangkat bergerak *non-telephony*

Termasuk dalam kategori ini adalah MP3 *Player* yang didominasi oleh iPod serta *video-enable* MP3 *Player* yang juga didominasi oleh *video* iPod. Dalam term *mobile learning*, MP3 *Player* yang dapat dimasukkan sebagai media *mobile learning* adalah perangkat MP3 *Player* yang memiliki koneksi ke internet.

2.1.2. Arsitektur Umum *Mobile Learning*

Agar tercapai sebuah aplikasi *mobile learning* yang dapat memenuhi kebutuhan sesuai dengan karakteristik pendidikan serta tuntutan mobilitas siswa, Yuni [4] mengungkapkan arsitektur umum *mobile learning* seperti pada Gambar 2.1.



Gambar 2.1. Aristektur Umum *Mobile Learning* [4]

Dalam penjelasannya, terdapat sebuah *mobile adaptor* yang diperlukan untuk memungkinkan adaptasi dari *e-learning* tradisional ke model yang dapat diterima oleh perangkat bergerak. Dalam Gambar 2.1, *mobile adaptor* berupa modul mLMS. Ada tiga modul utama dari sebuah mLMS, yaitu *Context Discovery*, *Mobile Content Management and Presentation Adaptation* dan *Packaging and Synchronization*.

Modul *Context Discovery* diperlukan untuk memperoleh informasi awal tentang perangkat yang digunakan *client* dalam mengakses materi *mobile*

learning. Informasi tentang perangkat yang dibutuhkan misalnya kesediaan sumber daya dari perangkat bergerak (resolusi layar, baterai, *bandwidth*, dan lain lain), aplikasi *browser* yang digunakan dan juga termasuk lokasi dari *client* itu sendiri. Masing-masing informasi ini dapat diperoleh saat *client* mengirimkan permintaan akses pertama kali, atau juga dapat disediakan saat dibutuhkan.

Modul *Mobile Content Management and Presentation Adaption* menjalankan fungsi yang penting dari arsitektur umum *mobile learning*. Presentasi dari materi pembelajaran merupakan isu yang penting dan harus didesign dengan sebaik-baiknya. Sebagai contoh, materi belajar yang diakses oleh PC tentu tidak akan mengalami masalah kompatibilitas. Namun bila materi belajar diakses oleh perangkat bergerak yang memiliki sejumlah keterbatasan, tentu akan timbul masalah kompatibilitas dari elemen-elemen yang tidak didukung oleh perangkat bergerak tersebut. Disini, modul *mobile content management and presentation adaption* memainkan peranan untuk menyesuaikan presentasi materi pembelajaran yang disesuaikan dengan informasi yang didapat dari *context discovery*.

Modul *Packaging and Syncrhonization*, berfungsi untuk memungkinkan siswa mengakses materi pembelajaran secara *offline*. Dengan modul ini, *client* yang memiliki keterbatasan akses koneksi internet, dapat melakukan pengunduhan materi yang diinginkan serta dapat membuka materi tersebut pada perangkat yang lain.

2.2. Aplikasi Web

Dalam *software engineering*, aplikasi web adalah sebuah aplikasi yang diakses dengan menggunakan aplikasi *web browser* melalui sebuah jaringan seperti internet dan atau intranet. Menggunakan aplikasi web, memiliki beberapa keuntungan yaitu

- Aplikasi *browser* pada umumnya membutuhkan ruang harddisk yang kecil pada *client*.
- Aplikasi dapat ter *upgrade* secara otomatis.
- Dapat diintegrasi dengan prosedur web lain dengan mudah, seperti email dan *search engine*.

- Bersifat *platform independent*, tidak terikat pada satu platform seperti halnya *standalone* aplikasi.

Sebuah aplikasi web terdiri dari komponen-komponen : *Markup Language*, *Web Server*, *Web Browser* dan *application-layer protocol*. Komponen-komponen tersebut akan dibahas berikut

2.2.1. *Markup Language*

Markup Language adalah sekumpulan anotasi teks yang menggambarkan bagaimana sturktur teks, tampilan dan format dari teks tersebut. *Markup Language* bisa dalam bentuk *manuscript form* atau dapat juga berupa *code* yang digunakan oleh sistem pengolah data pada sistem komputer. Salah satu contoh *markup language* yang cukup dikenal adalah *HyperText Markup Language* atau biasa disingkat menjadi HTML. HTML merupakan salah satu standar yang dihasilkan oleh sebuah konsorsium web internasional yang terdiri dari berbagai organisasi, staff dan relawan kerja untuk membuat standar teknologi web. Konsorsium ini dinamakan *World Wide Web Consortium* atau biasa disingkat menjadi W3C.Misi dari W3C ini adalah untuk mengarahkan kemampuan *World Wide Web* (WWW) ke potensi maksimal dengan membangun protokol dan panduan yang menjamin pertumbuhan jangka panjang dari WWW.

HTML digunakan oleh pengembang web, untuk mengatur tampilan dari aplikasi web yang dikembangkan. Selain HTML, W3C telah mengeluarkan standar *markup language* yang dapat digunakan untuk mengembangkan aplikasi web seperti XML, XHTML, WML dan masih banyak standar yang lain.

2.2.2. *Web Server*

Web Server adalah kombinasi dari komputer dan program yang terpasang pada komputer tersebut. *Web server* berinteraksi dengan *client* menggunakan aplikasi *web browser*. *Web server* mengirim halaman web ke komputer *client* dengan menggunakan protokol HTTP. Sebuah *web server* adalah program yang bersifat pasif, yaitu hanya menunggu dan baru memberikan respon jika ada permintaan akses dari *client*. Pada *web server*, terdapat sebuah program yang berfungsi untuk memantau adanya permintaan koneksi TCP dari *client* secara terus menerus. Jika ada permintaan koneksi TCP dari *client*, *web server* akan memberikan respon dan membuka sebuah koneksi. Setelah koneksi siap, *client*

akan mengirimkan *response message*. Selanjutnya koneksi akan kembali dibebaskan.

2.2.2.1. Server Side Programming

Pada *web server*, terdapat berkas HTML yang biasanya diminta oleh aplikasi *browser* untuk ditampilkan ke *client*. Berkas HTML seperti ini sifatnya adalah statis yang isinya tidak dapat diubah oleh *client*, kecuali oleh pengembangnya dengan mengubah teks *markup language* langsung ke berkas HTML. Agar berkas HTML menjadi dinamis, di mana isi dari HTML dapat berubah sesuai dengan kebutuhan *client*, dibutuhkan metode *programming* pada sisi *web server*.

Awalnya, *programming* ini dilakukan dengan menggunakan standar komunikasi antara aplikasi dan *web server* berupa sebuah aplikasi protokol *Common Gateway Interface* (CGI). Karena CGI adalah sebuah protokol, maka CGI dapat dibangun dengan menggunakan berbagai pemrograman yang ada, seperti C, C++, Perl, Java atau Visual Basic. Pada perkembangannya, penggunaan CGI terasa tidak efisien karena mekanisme CGI yang menciptakan satu proses untuk setiap permintaan *client* (*client request*) dan setelah selesai, proses tersebut akan langsung di akhiri.

Untuk mengatasi kelemahan CGI, Sun membangun *Servlets*, yaitu bagian dari aplikasi Java pada sisi *server* yang berfungsi untuk menjawab permintaan *client* melalui HTTP. Berbeda dengan CGI, *Servlets* dimuat ke memori pada saat pertama kali dipanggil dan tetap ada di memori hingga *server* di matikan.

Pengembangan web dengan menggunakan CGI dan *Servlets*, bagi sebagian pengembang ternyata di rasa masih banyak kendala. Dari kendala ini, muncul metode baru dalam mengembangkan aplikasi web. Metode ini disebut *Embedded Scripting Language*. Dengan metode ini, bahasa program dimasukkan ke dalam aplikasi *web server* yang kemudian akan mengolah setiap permintaan yang datang dari *client*. Yang termasuk dalam *Embedded Scripting Language* adalah *Java Server Pages* (JSP), *Active Server Pages* (ASP), dan *PHP Hypertext Preprocessor* (PHP). JSP adalah bahasa pemrograman *web* yang berbasis Java. JSP memiliki kemampuan untuk memanggil obyek bernama *Java Bean* yang mendukung pemisahan antara presentasi dan logik.

Sedangkan ASP dibangun oleh Microsoft dan merupakan *script* yang dapat ditulis dengan Perl, Jscript, dan atau VBScript. Pada platform Windows, ASP dapat memanggil *Componen Object Model* (COM) dari Microsoft, sebagaimana halnya JSP memanggil Java Bean. Selain itu ASP juga memudahkan pembangunan aplikasi basis data dengan menggunakan *Active Data Object* (ADO) dan SQL.

PHP merupakan *embedded scripting language* yang bersifat *open source*. PHP dikembangkan berbasis bahasa C. PHP mendukung SQL sehingga dapat digunakan untuk membangun aplikasi berbasis data. Selain itu, PHP juga mudah diintegrasi ke dalam *markup language* sehingga saat ini, banyak pengembang web menggunakan PHP sebagai bahasa pemrogramannya. PHP memiliki modul-modul yang mendukung penggunaan PHP secara luas.

2.2.3. *Web Browser*

Web Browser adalah sebuah aplikasi untuk menerima, mengubah dan menampilkan *markup language* (biasanya berupa HTML) yang dikirim *web server* ke hadapan *client*. Aplikasi *web browser* biasanya digunakan untuk mengakses WWW baik yang melalui jaringan Internet maupun Intranet.

Proses untuk menampilkan halaman HTML yang dikirim dari *web server*, dimulai dari *client* yang memasukkan *Uniform Resource Identifier* (URI) ke dalam aplikasi *web browser*. Awalan URI menentukan bagaimana URI akan diperlakukan. Biasanya, URI dimulai dengan *http:* yang mengindikasikan bahwa materi akan diterima melalui *HypterText Transfer Protocol* (HTTP). Banyak *browser* juga mendukung bentuk awalan yang lain seperti *https:* untuk HTTPS, *ftp:* untuk *File Transfer Protocol*, dan *file:* untuk berkas lokal. Awalan yang tidak dapat ditangani langsung oleh *web browser* akan langsung diserahkan kepada aplikasi lain untuk menanganinya.

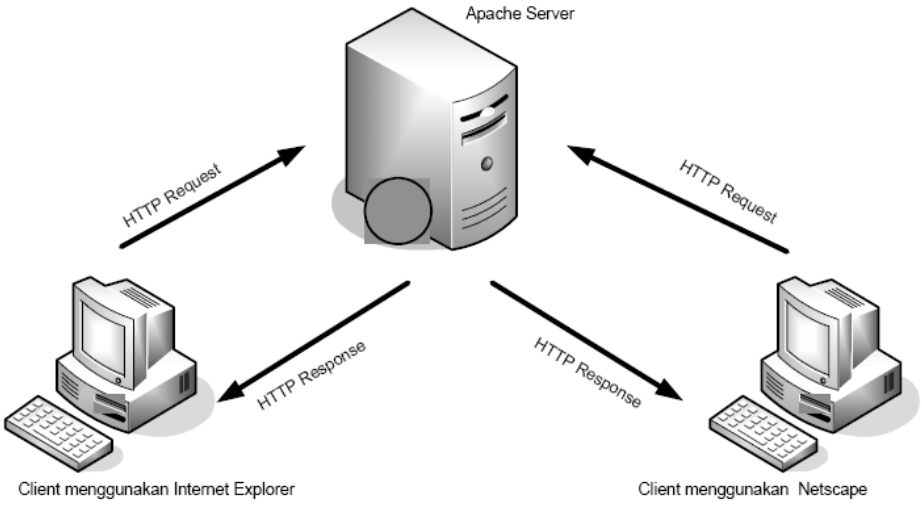
Aplikasi *web browser*, saat ini tidak hanya terpasang pada komputer PC maupun Laptop. Dengan meningkatnya kemampuan *processing* dari perangkat bergerak, saat ini sudah banyak *Mobile Phone* yang dilengkapi dengan aplikasi *browser* untuk mengakses *mobile web*. Pengembang *Opera Browser* juga telah mengembangkan *Opera Mini* untuk kebutuhan mengakses *web* menggunakan

perangkat bergerak. Dengan adanya fasilitas ini, aplikasi *web* kembali mengalami kemajuan untuk menyentuh kalangan pengguna yang lebih luas.

2.2.4. *Application-Layer Protocol*: HTTP

HyperText Transfer Protocol atau HTTP adalah protokol pada tingkat *application layer* untuk distribusi, kolaborasi sistem informasi multimedia. HTTP telah digunakan secara luas pada *World-Wide Web* sejak tahun 1990. Versi pertama dari HTTP yang dikenal sebagai HTTP/0.9, merupakan protokol yang sederhana, hanya mengirim data mentah melalui internet. HTTP/1.0 yang digambarkan pada RFC 1945, meningkatkan kemampuan protokol dengan mengijinkan sistem *message* dalam format MIME-like, mengandung *metainformation* mengenai data yang telah dikirim dan perubahan pada *request/response semantic*. Bagaimanapun, HTTP/1.0 belum cukup mempertimbangkan efek dari proxy yang berlapis, *caching*, kebutuhan untuk koneksi terus menerus atau *virtual host*. Sebagai akibat implementasi yang belum sempurna dari HTTP/1.0, dibutuhkan protokol baru untuk mengakomodir dua komunikasi antar aplikasi untuk menentukan kemampuan dari masing-masing.

Spesifikasi yang baru memunculkan protokol HTTP yang baru yaitu HTTP/1.1 [7]. Protokol ini memasukkan kebutuhan yang lebih keras dibanding HTTP/1.0 dengan tujuan untuk menyakinkan ketahanan implementasi dari fitur-fitur yang dimiliki.



Gambar 2.2. Interaksi *Web Server-Client via HTTP/1.1* [4]

Dari Gambar 2.2 dapat terlihat bagaimana sebuah *client* mengakses *web*. HTTP/1.1 menggunakan TCP sebagai protokol pada layer *Transport*. Mula-mula HTTP pada sisi *client* akan menginisiasi TCP *connection* dengan *server*. Ketika koneksi sudah terbentuk, *browser* dan *server* melakukan pengaksesan TCP melalui masing-masing *socket interface*. Koneksi yang digunakan adalah *persisten connection* dengan *pipelining*. Dengan jenis koneksi ini maka *server* akan membiarkan koneksi TCP tetap terbuka setelah mengirimkan sebuah *response message*, sehingga *request* dan *response* berikutnya di antara kedua *server* dan *client* yang sama dapat menggunakan koneksi tersebut. Hal ini menjadikan kinerja *server* lebih efisien, dibandingkan dengan mode *non persistent* di mana untuk setiap *request* yang dikirim harus dibangun sebuah koneksi baru. Sedangkan dengan mode *pipelining* maka *client* dapat mengirimkan *request* begitu mendapatkan referensi tentang obyek yang dituju. Dengan demikian untuk mengirimkan *request* berikutnya *client*tidak harus menunggu datangnya *response* dari *request* sebelumnya.

Aplikasi *browser* yang mendukung HTTP/1.1 mengirimkan informasi yang lebih detail. Ketika *client* mencoba untuk menerima sebuah halaman *web*, *browser* mengirim sebuah *request* ke *web server*. Informasi ini dikenal sebagai *Accept Header*. Versi HTTP/1.1 menggunakan empat header, yaitu Accept, Accept-Char-set, Accept-Encoding dan Accept-Language. Berikut adalah contoh header yang dihasilkan dari Mozilla Firefox 3.

```
GET *.* HTTP/1.1
Host: content.yieldmanager.edgesuite.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.11) Gecko/2009060215 Firefox/3.0.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://www.addictivetips.com/internet-tips/view-http-headers-of-any-web-page-in-firefox/
```

Dengan informasi yang ada pada *header*, menjadi bahan pertimbangan bagi *web server* untuk menghasilkan atau membuat halaman *web* yang sesuai dengan kemampuan dari *browser*.

2.3. Standarisasi *Mobile Web*

W3C bersama grup-grup penelitian telah membuat beberapa standarisasi berkaitan dengan pengembangan *mobile web* yang merupakan induk dari pengembangan aplikasi *mobile learning*. Standar ini sangat diperlukan baik oleh produsen perangkat bergerak maupun pengembang aplikasi *mobile web* untuk menjamin bahwa aplikasi *mobile web* yang dikembangkan dapat berjalan di semua perangkat bergerak yang ada dan tidak terpaku kepada salah satu perangkat bergerak saja. Dengan demikian, potensi *mobile web* dapat dikembangkan semaksimal mungkin.

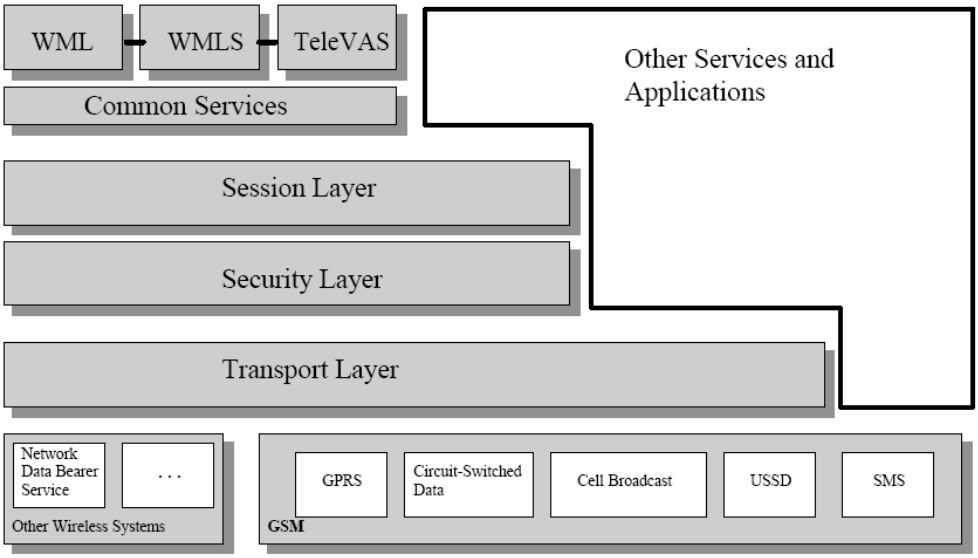
Standar yang dikembangkan oleh W3C untuk keperluan *mobile web* disebut *Wireless Application Protocol* versi 1 (WAP 1.0) dan *Wireless Application Protocol* versi 2 (WAP 2.0).

2.3.1. Pengenalan WAP 1.0

WAP merupakan sekumpulan standar yang pertama di luncurkan untuk mengakses *web* dari perangkat bergerak [8]. WAP sebenarnya merupakan sekumpulan standar yang mengkomodir format *markup language* (WML) dan protokol yang digunakan untuk melayani WML (WTP. WTLS, dan lain-lain). Kumpulan standar WAP diatur oleh *Open Mobile Alliance* (OMA).

Ada dua evolusi utama dari WAP. WAP 1.0 merupakan standar yang dominan di masa awal pengenalan *mobile web*. Hampir semua penyedia layanan bergerak dan produsen perangkat bergerak di Eropa dan Amerika Utara tetap mendukung WAP 1.0.

Gambar arsitektur dari WAP 1.0 seperti pada Gambar 2.3.



Gambar 2.3. Arsitektur WAP 1.0 [8]

Arsitektur WAP 1.0 menyediakan lingkungan yang bisa dikembangkan untuk pengembangan aplikasi pada perangkat komunikasi bergerak. Hal ini dicapai melalui design berlapis dari jaringan hingga *application communication protocol stack*. Setiap lapis dari arsitektur dapat di akses melalui lapis di atasnya sama baiknya dengan layanan dan aplikasi yang lain.

Arsitektur WAP 1.0 memungkinkan berbagai layanan dan aplikasi yang disesuaikan dengan fitur dari *stack* WAP. Aplikasi yang dispesifikasi dalam *framework* WAP 1.0 memungkinkan pemanfaatan layanan biasa dari lapis internal hingga lapis aplikasi sementara aplikasi eksternal bisa mengakses lapis *session* dan *transport* secara langsung. Akses langsung dari aplikasi eksternal ke lapis *security* masih bersifat opsional.

Komponen utama yang dibutuhkan oleh sistem WAP seperti yang ditunjukkan pada Gambar 2.3. adalah :

1. WML (*Wireless Mark-up Language*) *Browser*
2. WMLScript *Interpreter*
3. TeleVAS *features*
4. *The Session Protocol Layer* (WSP)
5. *A Security Protocol Layer*
6. *The Transport Protocol Layer* (WTP)
7. *Content Formats*

Ruang lingkup WAP melingkupi dari lapis *transport* hingga lapis *application* dari susunan *stack* WAP.

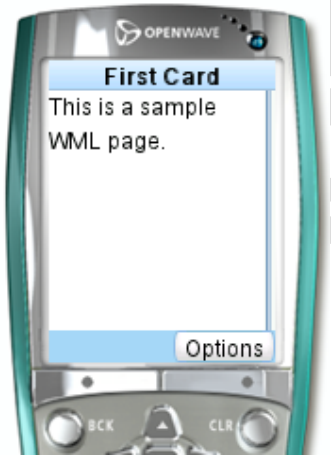
Pada WAP 1.0, WML menjadi *markup language* yang utama. WML adalah *markup language* yang berbasis XML yang sangat berbeda dari HTML (*markup language* yang biasa digunakan pada aplikasi *web*). WML didukung oleh hampir semua produsen *mobile phone*. WML di design untuk mendukung perangkat bergerak yang memiliki karakteristik berikut :

- Ukuran layar yang kecil (dibandingkan layar PC yang umum)
- *Memory* dan ukuran CPU yang terbatas.
- *Bandwidth* rendah dengan *high-latency wireless connectivity*.

Perangkat yang sekarang mendukung WML dibagi menjadi dua kategori utama yaitu :

- *Mobile phone*, yang memiliki ciri-ciri teks layar dengan *range* dari 4 hingga 10 baris dan mendukung *user input* melalui tombol angka dan fungsi.
- *Personal Digital Assistants* (PDAs), yang memiliki ciri-ciri resolusi layar dari 100x100 pixel dan mendukung penambahan *user input* melalui *keypad*, *pointer* dan pengenalan tulis tangan.

Contoh script dari WML adalah seperti berikut :




```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML
1.1//EN""http://www.phone.com/dtd/wml11.dtd"
>
<wml>
<card id="main" title="First Card">
<p mode="wrap">This is a sample WML page.</p>
</card>
</wml>
```

Gambar 2.4. Contoh WML

WML telah didukung secara luas di hampir semua *browser* yang terpasang di *mobile phone* keluaran terbaru. Namun kekurangan dari WML adalah

keterbatasan kemampuan design yang membatasi kreatifitas dari pengembang aplikasi *mobile web* serta tidak mendukung fitur-fitur terbaru yang kini mulai dimiliki oleh *mobile phone* modern.

2.3.2. Pengenalan WAP 2.0

WAP 2.0 adalah standar *mobile web* saat ini. Satu tujuan utama dari WAP 2.0 adalah untuk meningkatkan kemampuan perangkat bergerak mendekati kemampuan komputer PC dengan mengadopsi beberapa perubahan [9] :

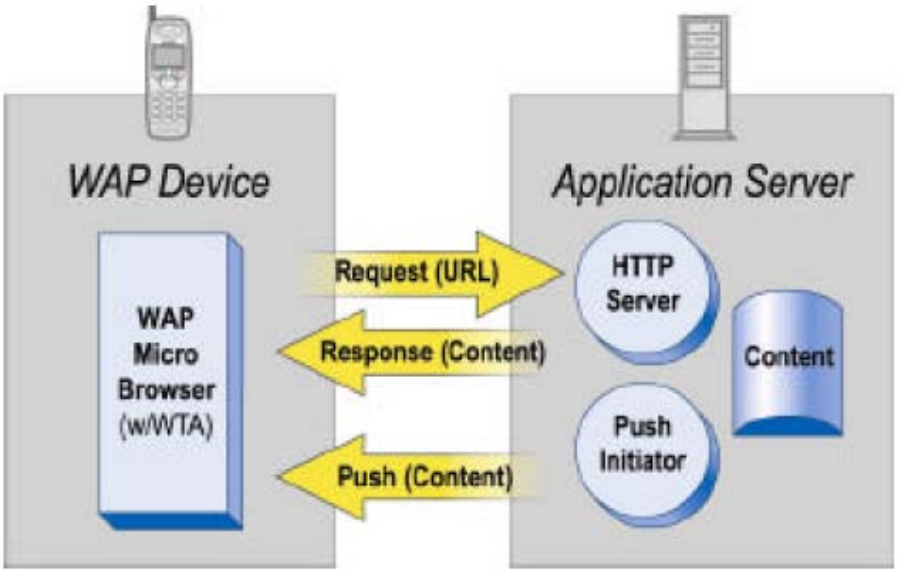
- Mendukung standar protokol komunikasi internet seperti TCP/IP dan HTTP, tidak hanya protokol yang sudah dikenal oleh WAP 1.0.
- Mengadopsi XHMTL-MP sebagai *markup language* yang utama.

Model *programming* WAP 2.0 hampir sama dengan model *programming* WWW dengan beberapa pengembangan. Mengadopsi model *programming* WWW, menawarkan beberapa keuntungan bagi komunitas pengembang aplikasi termasuk yang telah mengenal model *programming*, arsitektur yang handal dan kemampuan untuk menggunakan alat yang sudah ada seperti *web server*, XML dan lain-lain. Optimasi dan penambahan telah dibuat untuk menyesuaikan standar dengan karakteristik lingkungan *wireless*. Jika mungkin, standar yang sudah ada telah diadopsi atau telah digunakan sebagai titik awal dari teknologi WAP.

Pengembangan WAP 2.0 yang paling signifikan adalah :

- *Push*
- *Telephony Support* (WTA)

Model *programming* WAP seperti pada Gambar 2.5.



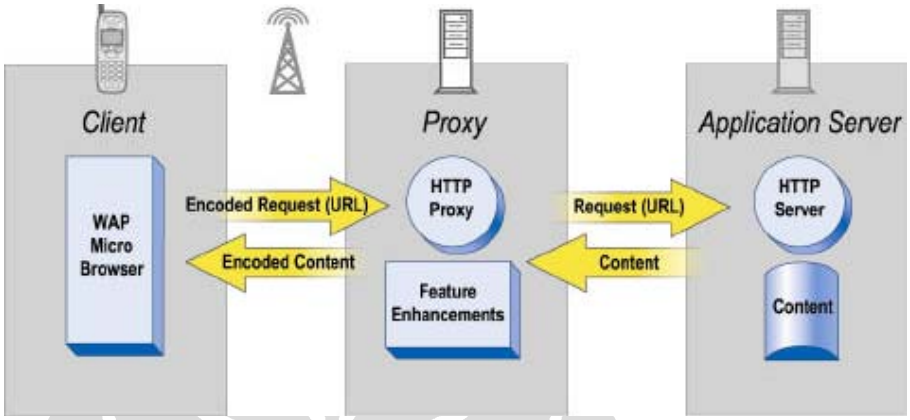
Gambar 2.5. Model *Programming* WAP 2.0 [10]

Materi WAP 2.0 dan aplikasi ditampilkan dalam sekumpulan format materi yang telah dikenal berdasar pada format materi WWW. Materi di kirim menggunakan protokol komunikasi yang standar berdasar pada protokol komunikasi WWW. *Microbrowser* WAP mendefinisikan sekumpulan komponen standar yang memungkinkan komunikasi antara perangkat bergerak dengan *server* jaringan. Komponen tersebut antara lain [10] :

- *Standard naming model*, URL standar WWW digunakan untuk mengidentifikasi materi WAP pada *server* asal. URI standar WWW digunakan untuk mengenali sumber daya pada perangkat seperti *call control function*.
- *Content typing*, semua materi WAP 2.0 memiliki tipe yang spesifik yang konsisten dengan WWW *typing*. Hal ini mengijinkan *user agent* WAP 2.0 untuk memproses materi secara benar berdasar tipe-nya.
- *Standard content formats*, format materi WAP 2.0 berdasarkan teknologi WWW dan termasuk *markup display*, kalender informasi, dan *scripting language*.
- *Standard communication protocols*, protokol komunikasi WAP 2.0 memungkinkan permintaan komunikasi *browser* dari terminal *mobile* ke jaringan *web server*.

Tipe materi dan protokol WAP 2.0 telah dioptimasi untuk pasar yang luas dan mendukung perangkat *hand-held wireless*.

Feature/Performance-Enhancing Proxies



Gambar 2.6. Proxy Peningkatan Fitur/Performa WAP 2.0 [10]

WAP 2.0 menggunakan teknologi proxy untuk mengoptimasi dan meningkatkan koneksi antara domain *wireless* dan WWW. *Proxy* WAP 2.0 memungkinkan berbagai fungsi, termasuk di antaranya :

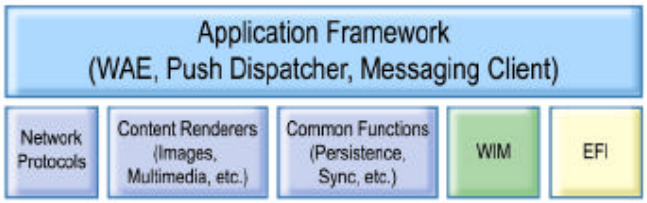
- *Protocol Gateway*, mengubah permintaan dari sebuah *stack wireless protocol* (misal WAP 1.0 *stack* seperti WSP, WTP, WTLS, dan WDP) ke protokol WWW (HTTP dan TCP/IP). *Gateway* juga melakukan DNS *lookups* dari *server* yang terdapat pada URL.
- *Content Encoders and Decoders*, *encoder* materi bisa digunakan untuk mengubah materi WAP 2.0 ke dalam format yang lebih sederhana yang memungkinkan utilisasi yang lebih baik dari halmaan yang dituju karena ukurannya yang sudah dikurangi.
- *User Agent Profile Management*, menggambarkan kemampuan *client* dan *personal preference* (UAProf) di kemas dan disampaikan ke aplikasi.
- *Caching Proxy*, untuk meningkatkan performa penerimaan dan utlisasi jaringan dengan mengatur cache dari sumber daya yang sering di akses.

Infrastruktur ini memungkinkan pengguna perangkat bergerak untuk mengakses berbagai materi internet dan aplikasi, dan administrator aplikasi dapat membangun materi layanan dan aplikasi yang dapat berjalan pada perangkat bergerak yang lebih luas. *Proxy* WAP 2.0 memungkinkan materi dan aplikasi

untuk bisa disimpan pada *web server* berbasis WWW dan dikembangkan menggunakan teknologi WWW yang sudah biasa dipakai seperti CGI *scripting*.

Sementara jumlah penggunaan WAP 2.0 akan memasukkan sebuah *web server*, WAP *proxy* dan WAP *client*, WAP 2.0 arsitektur dengan mudah mendukung konfigurasi yang lain.

Arsitektur *client* WAP 2.0



Gambar 2.7. Arsitektur WAP *client* [10]

Arsitektur untuk perangkat WAP 2.0 seperti ditunjukkan pada Gambar 2.7. *Framework* aplikasi memungkinkan lingkungan perangkat untuk mengeksekusi aplikasi WAP 2.0. Aplikasi WAP 2.0 terdiri dari *markup language*, *script*, *style sheets* dan materi multimedia yang semuanya diserahkan kepada perangkat. WAP *Application Environment* (WAE) mengolah model yang menggambarkan struktur dimana berbagai bentuk format dari materi *executable* dan *non-executable* saling berinteraksi.

Protokol jaringan pada WAP *client* di bagi antara *client* dan *server*. Materi di interpretasi ke dalam format materi yang spesifik dan menyediakan materi tersebut kepada pengguna akhir untuk interaksi. Fungsi umum diutilisasi oleh *framework* aplikasi, termasuk persistensi dan sinkronisasi data.

Wireless Indetity Module (WIM) mengandung identitas dari perangkat dan kriptografi yang memungkinkan autentikasi antara perangkat WAP 2.0 dan *server*. Arsitektur juga memungkinkan mekanisme untuk mengakses fungsi eksternal yang terpasang pada perangkat melalui *External Functionality Interface* (EFI).

Untuk mengatasi kekurangan WML yang menjadi *markup language* utama pada WAP 1.0, WAP 2.0 menggunakan XHTML-MP sebagai *markup language* utama. XHTML-MP (*Extensible Hypertext Markup Language – Mobile Profile*) adalah XHTML yang khusus didesign untuk bekerja sama dengan fitur yang

terdapat pada perangkat bergerak. XHTML-MP 1.0 didefinisi oleh OMA dan merupakan pengembangan dari XHTML Basic 1.0 yang dibuat oleh W3C.

Karena XHTML Basic dan XHTML-MP adalah sub bagian dari XHTML, tarnsisi bagi pengembang untuk menghasilkan materi yang sesuai untuk perangkat bergerak tidak terlalu sulit. Alat yang biasa digunakan seperti aplikasi *desktop browser* dan *editor* bisa digunakan untuk membangun aplikasi *mobile* dan pengembang tidak perlu mempelajari *markup language* yang baru.

Wireless CSS

XHTML-MP hadir bersama Wireless CSS untuk memisahkan sisi presentasi dari *markup language*, sama seperti pada *desktop*. OMA mengatur standar *Wireless CSS* sebagai sub bagian dari CSS dan juga bagian dari spesifikasi WAP 2.0. Menjadi catatan bahwa *Wireless CSS* tidak compatible dengan WML karena keterbatasan yang dimiliki WML.

Wireless CSS dapat ditambahkan pada *markup language* XHTML-MP sama halnya pada dokumen HTML biasa. Berikut adalah contoh *script* untuk menggunakan *stylesheet* eksternal :

```
<link href="external.css" rel="stylesheet" type="text/css" />
```

Berikut adalah contoh script *Wireless CSS* :

```
<style>
p {
font-size: small;
}
</style>
```

Wireless CSS mendukung banyak atribut CSS, namun tidak seluruhnya. Menjadi catatan bagi pengembang bahwa teknik lanjutan dalam mendesign halaman menggunakan CSS belum tentu dapat berjalan pada berbagai jenis *mobile browser*. Sehingga dianjurkan untuk menggunakan *wireless css* secara sederhana.

2.4. Kemampuan Perangkat Mobile

Untuk dapat menyampaikan aplikasi *web* secara maksimal, kemampuan dari perangkat yang digunakan untuk menampilkan aplikasi *web* sangat perlu dipertimbangkan. Kemampuan perangkat yang perlu dipertimbangkan antara lain : resolusi tampilan, dukungan teknologi *web*, memori serta kecepatan processor. Untuk mempertimbangkan perangkat bergerak sebagai alat mengakses aplikasi

web, tidak ada salahnya melakukan perbandingan dengan kemampuan PC pada 10 tahun yang lalu ketika aplikasi *web* mulai diperkenalkan pada komputer PC. Berikut adalah perbandingan spesifikasi komputer PC 10 tahun yang lalu dengan kemampuan perangkat bergerak dewasa ini [11]:

- **Kapasitas penyimpanan**

  Pada tahun 1998-an, kapasitas *harddisk* untuk PC yang biasa digunakan sekitar 2GB. Saat ini, kapasitas memori *flashdisk* yang biasa digunakan juga sekitar 2GB.

- **Kecepatan *Processing***

  Intel mengeluarkan *desktop* CPU 300MHz – 700MHz di tahun 1998 dan saat ini mengeluarkan *XScale processor* dengan kecepatan yang sama untuk perangkat bergerak di tahun 2005.

- **Ukuran Layar**

  Di tahun 1995, rata-rata ukuran monitor PC adalah 14 inci dengan ukuran layar 640x480 pixel (XGA). DI tahun 2005, PDA seperti Dell X50 mengeluarkan produk dengan resolusi 480x640 (walaupun masih banyak yang mengeluarkan produk dengan ukuran layar 420x480 atau kurang).

- ***Bandwidth* yang terbatas**

  Pada tahun 1996, banyak komputer PC yang saling terhubung dengan kecepatan yang rendah (sekitar 10Mbps). Aksses ke internet menggunakan 28.8kpbs dial-up modem dan terasa sangat mahal untuk ukuran sekarang. Saat ini, beberapa perangkat bergerak telah memiliki kemampuan koneksi data tetapi dengan keandalan jaringan yang belum memadai dan daerah cakupan yang belum luas.

Dari perbandingan kemampuan perangkat di atas, dapat disadari bahwa panduan untuk membangun *web* pada 10 tahun yang lalu untuk komputer PC, untuk mengakomodir keterbatasan yang ada, bisa langsung diterapkan untuk menggunakan perangkat bergerak sebagai media *mobile web* terutama *mobile learning*.

Untuk menampilkan halaman *web* yang didesign untuk ukuran layar komputer PC, tentu saja memerlukan penyesuaian sebelum ditampilkan pada ukuran layar perangkat bergerak. Tanpa ada penyesuaian, tampilan dari halaman

web pada perangkat bergerak, tidak senyaman pada layar komputer PC. Demikian juga dengan jenis teknologi yang didukung. Dengan adanya dukungan *scripting* yang banyak pada komputer PC (misalnya Javascript, Flash, dan lain-lain) membantu pengembang untuk dapat berkreasi mengatur tampilan *web*. Pada perangkat bergerak yang masih terbatas dukungan *scripting*, tentu akan ada beberapa *scripting* yang tidak dapat dijalankan pada perangkat bergerak sehingga kemampuan aplikasi *web* yang ditampilkan tidak maksimal.

Session dan *Cookies* juga menjadi perhatian bagi pengembang *web* dalam membangun aplikasi *web* untuk perangkat bergerak. *Session* dan *Cookie*s merupakan salah satu fitur yang sangat dibutuhkan untuk membangun aplikasi yang membutuhkan data diri pengguna untuk mengatur tampilan aplikasi *web* yang sesuai dengan preferensi pengguna. *Session* dan *cookies* juga dimanfaatkan pada aplikasi *web* yang menerapkan sistem login. Saat ini, tidak semua aplikasi *browser* yang terpasang pada *mobile phone* memiliki dukungan terhadap *session* dan *cookies*. Sehingga membutuhkan trik tersendiri untuk dapat membangun aplikasi seperti yang telah diterangkan sebelumnya.

Agar tampilan dapat disesuaikan dengan tepat terhadap perangkat yang mengakses aplikasi *web* tersebut, diperlukan informasi yang dapat memberikan gambaran tentang kemampuan dari sebuah perangkat. Pada bagian *application-layer protocol* telah dibahas tentang *HTTP header* pada teknologi HTTP/1.1 yang dapat memberikan beberapa informasi tentang hal tersebut.

2.4.1. *Composite Capability / Preference Profile*

Composite Capability/Preference Profile (CCPP) adalah standar yang dikeluarkan oleh W3C, yang berfungsi untuk mengkomunikasikan kemampuan perangkat seperti *client*, *proxy*, *gateway*, *cache* juga sumber daya (misalnya dokumen). CC/PP didasarkan pada *Resource Description Framework* (RFD), yaitu standar yang juga dikeluarkan oleh W3C untuk mengidentifikasi sebuah halaman *web* sehingga memudahkan indeks oleh mesin pencari (*search engine*).

Beberapa identifikasi yang dapat diberikan oleh CC/PP adalah *print and display devices*, *user agent headers*, dan *proxy servers*. *Print and display devices* memberikan informasi tentang identitas perangkat, ukuran layar (dalam karakter dan *pixel*), tipe MIME, tipe-tipe karakter, serta *color palette*. *User agent headers*

memberikan rincian tentang CPU, ukuran layar, nama sistem operasi dan versinya, nama *browser* dan versinya, serta tipe MIME yang dapat diterima. Sedangkan *proxy server* memberikan detil tentang tipe MIME yang dapat diterima, ditransformasikan dan ditolak oleh *client*.

2.4.2. *WAP User Agent Profile*

WAP User Agent Profile (UAProf) adalah semacam CC/PP yang dikeluarkan oleh WAP forum untuk memungkinkan perangkat berbasis WAP dideklarasikan kemampuannya pada saat menghubungi *server*. Dengan UAProf, sebuah perangkat berbasis WAP dapat memberikan identitas dirinya pada saat menghubungi sebuah *server*. UAProf memberikan detil informasi yang berhubungan dengan *software, hardware, browser, network,* dan WAP. Beberapa informasi yang dapat diberikan oleh UAProf diantaranya adalah : nama sistem operasi dan versi-nya, nama *browser* dan versinya, versi HTML dan XHTML, CPU, ukuran layar, resolusi, serta versi WAP dan WML.

2.5. Teknik Adaptasi *Web*

Sebuah proses adaptasi diperlukan untuk menyesuaikan sebuah halaman Web dengan perangkat dan browser yang mengaksesnya. Beberapa proses adaptasi yang ada adalah: *select*/*remove*, adaptasi navigasi, substitusi, dan transformasi.

Pada adaptasi *select*/*remove* adaptasi web dilakukan dengan cara menentukan kelompok-kelompok menu yang dapat diakses oleh suatu perangkat/browser, atau mengarahkan suatu permintaan dari client ke server yang sesuai dengan perangkat/browser yang digunakan client (*URL redirection*). Cara yang kedua ini umumnya dilakukan oleh server proxy yang bertindak sebagai *redirecting proxy*.

Adaptasi navigasi menggunakan pembangkit navigasi (*navigation generator*) untuk mendapatkan struktur navigasi yang spesifik untuk setiap jenis perangkat *client* berdasarkan konten/data yang dimiliki oleh sebuah web. Struktur navigasi yang umum digunakan adalah dengan membuat daftar isi (*table of content*), menu navigasi, dan alur *next*/*previous*.

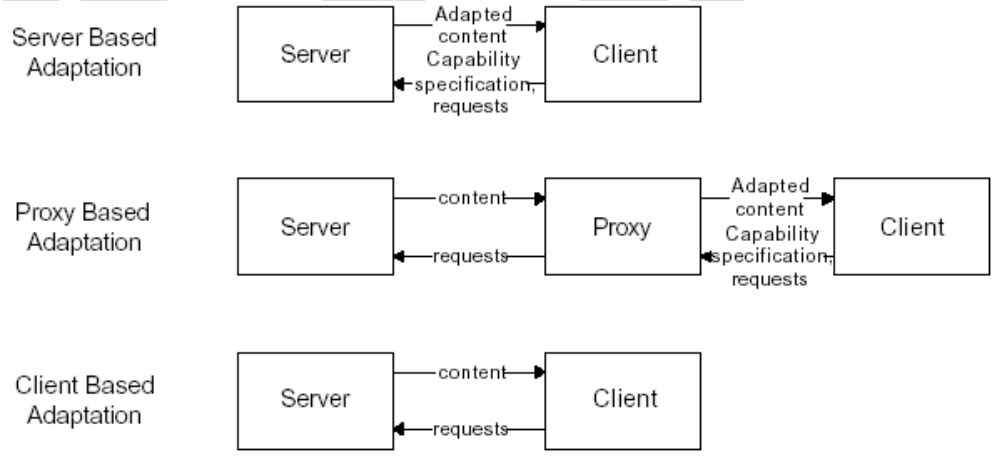
Proses adaptasi yang ketiga adalah substitusi, yaitu mengganti sebuah isi/informasi yang muncul secara berulang pada beberapa halaman *web* dengan

sebuah *link* yang menuju ke informasi tersebut. Dengan demikian informasi lengkap hanya akan muncul jika *link* tersebut diakses. Cara ini digunakan terutama pada perangkat yang memiliki keterbatasan *bandwidth*, memori dan ukuran layar.

Proses adaptasi yang terakhir adalah transformasi, yaitu adaptasi yang melibatkan konversi dari suatu format/markup language ke markup language lainnya. Proses adaptasi ini selain melibatkan XSLT dan DOM untuk markup transcoding atau prosesor transformasi, juga dapat dilakukan dengan menggunakan arsitektur aplikasi *Model-View-Controller* pada *server side programming*.

2.5.1. Arsitektur Umum Adaptasi

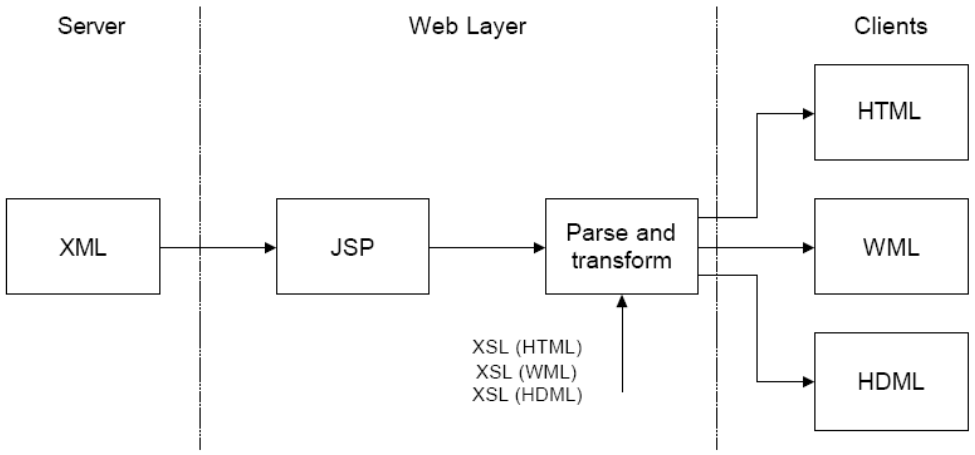
Berdasarkan lokasi dimana proses adaptasi terjadi, maka arsitektur adaptasi dapat dibedakan menjadi tiga yaitu server-based, proxy-based, dan client-based. Ketiga arsitektur adaptasi tersebut ditunjukkan pada Gambar 8 [3].



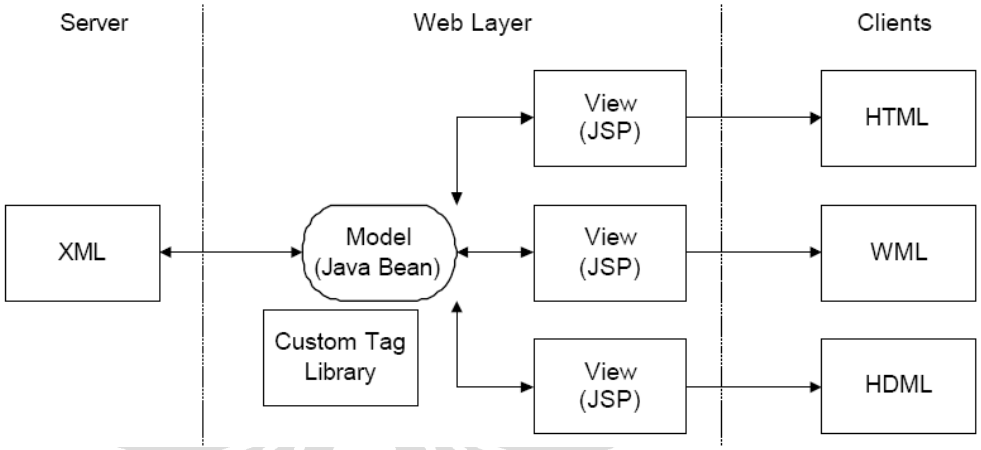
Gambar 2.8. Tiga Tipe Adaptasi [3]

Adaptasi pada *server* dapat dilakukan dalam berbagai format. Salah satu format adaptasi yang dilakukan pada *server* adalah transformasi XML dengan XSLT. Transformasi XML dengan XSLT dilakukan dengan tiga pendekatan, yaitu *single pipeline*, *multiple pipeline* dan *combination*. Pada pendekatan *single pipeline* seperti pada Gambar 9 [12], *server side programming* akan menghasilkan XML yang kemudian diubah menjadi banyak *markup language* yang ditentukan oleh XSLT *stylesheet*. Sebaliknya, pendekatan pada Gambar 10 [12], terdapat

banyak *script* yang tergantung kepada jumlah perangkat tetapi dapat menggunakan modul maupun sumber daya yang ada.



Gambar 2.9. *Single Pipeline* [12]



Gambar 2.10. *Multiple Pipeline* [12]

Cara yang lain adalah dengan menggunakan *scripting* pada bagian terkecil dari pengiriman halaman. Salah satu contoh adalah dengan mengubah *HTTP Content-Type header* yang terpasang pada isi. Adaptasi dengan cara ini membutuhkan sebuah pengujian untuk mengidentifikasi jenis perangkat yang mengakses aplikasi *web* dan mengirim *header* yang sesuah dengan jenis perangkat tersebut.

Berikut adalah contoh mudah dalam menguji jenis perangkat dengan menggunakan bahasa pemrograman PHP.

```
<?php
$msie = strpos($_SERVER["HTTP_USER_AGENT"],'MSIE') > 0;
if (!$msie) {
  header("Content-Type:
  application/xhtml+xml;
  charset=UTF-8");}
else {
  header("Content-Type: text/html;
  charset=UTF-8");}
?>
```

Gambar 2.11. Contoh *script* pengenalan *User Agent* [9]

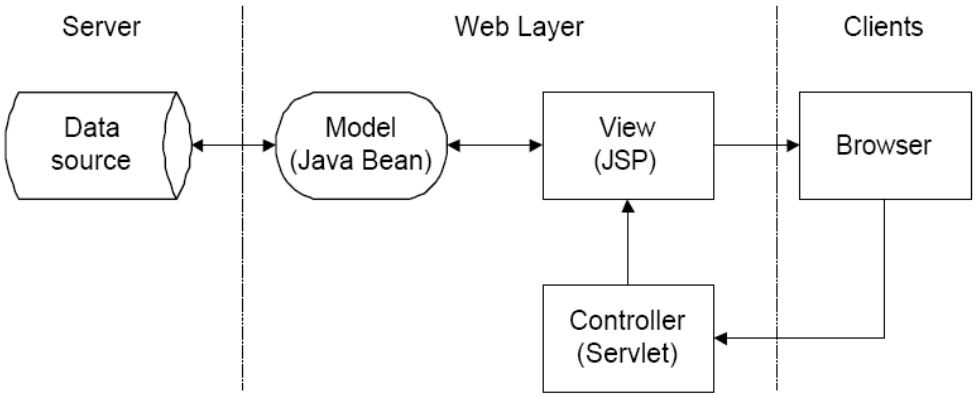
Contoh yang lain untuk menyesuaikan halaman web dengan karakteristik perangkat adalah penyesuaian ukuran gambar yang sesuai dengan perangkat. Sebagai panduan, perangkat bergerak *low-end* membatasi gambar dengan ukuran lebar 120px. Sementara yang *high-end* akan memiliki ukuran gambar yang lebih baik. Penyesuaian ukuran gambar dirasa perlu untuk menghemat *bandwidth* yang digunakan untuk mengirimkan gambar. Hal ini dilakukan dengan melakukan perubahan ukuran gambar pada saat transmisi ke *server*. Teknik penyesuaian gambar dapat dilakukan dengan memilih ukuran dengan prediksi ukuran terbesar yang dapat diakses oleh perangkat atau melakukan perubahan secara dinamis pada saat pengiriman ke perangkat.

Adaptasi pada sisi *client* biasanya tergantung kepada dukungan standar pada perangkat yang digunakan. Adaptasi pada *client* dilakukan dengan menyesuaikan *stylesheet* yang digunakan untuk mengatur tampilan pada perangkat. Dengan standarisasi yang telah dikeluarkan oleh W3C, adaptasi pada sisi *client* bisa menjadi bentuk adaptasi yang paling berguna walau tetap terdapat keterbatasan. *Markup language* XHTML yang dihasilkan oleh *server* mungkin saja terdapat bagian yang tidak dapat ditampilkan oleh perangkat bergerak. Karena itu perlu dilakukan penyesuaian pada sisi *server* agar memenuhi standarisasi W3C untuk perangkat bergerak agar dapat ditampilkan dengan baik.

2.5.2. Arsitektur Aplikasi

Banyak aplikasi saat ini yang berbasis *three tier* di mana membagi aplikasi menjadi sumber data, *middle*, dan *client tier*. Bentuk aplikasi yang lain disebut *Model-View-Controller* untuk memisahkan bagian *presentation* (*View*) dengan

logik (*Model*) aplikasi. Model ini telah digunakan sejak tahun 1980. Model ini telah diidentifikasi menjadi model kunci untuk mengembangkan aplikasi *web*. Pada arsitektur MVC, untuk memasangkan data dari *model* dengan *presentation* (*view*) merupakan tugas dari *controller*. Model arsitektur MVC ditunjukkan pada Gambar 2.12. yang diasosiasikan dengan bahasa pemrograman Java 2 Enterprise Edition (J2EE), dan juga JSP, Servlets dan Java Beans walau hal ini juga mungkin diimplementasikan pada ASP atau PHP.



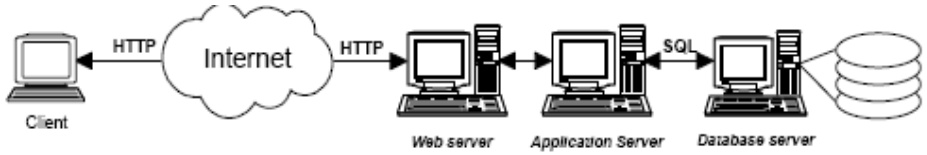
Gambar 2.12. Arsitektur *Model - View – Controller* [12]

2.6. *Dynamic Web Page*

Dynamic Web Page mulai diperkenalkan seiring dengan perkembangan *server side programming* seperti yang telah dijelaskan pada pembahasan sebelumnya. Untuk memahami pengertian *dynamic web page* tidak terlepas dari pengertian *static web page*. Pengertian *static web page* adalah sebuah halaman web yang hanya menampilkan informasi yang sama setiap kali halaman ini diakses oleh pengguna dari aplikasi *browser*. Perubahan dari informasi dalam *static web page* hanya dimungkinkan dengan mengubah langsung *source code* dari halaman tersebut.

Untuk memungkinkan halaman web yang menampilkan informasi yang sesuai dengan keinginan pengguna tanpa harus merubah *source code* dari halaman sebuah *web*, muncul istilah *dynamic web page* yang dimungkinkan dengan menggunakan *server side programming*. Sebuah *dynamic web page* mengandung informasi yang dapat berubah sesuai dengan kebutuhan pengguna yang mengakses *web* tersebut.

Umumnya isi dari *dynamic web page* dibangun dari kombinasi tiga komponen yaitu *web server, application server,* dan database. Kombinasi tiga komponen ini ditunjukkan pada Gambar 2.13.



Gambar 2.13. Konfigurasi *Dynamic Web Page* [15]

Informasi dari halaman disimpan dalam sebuah aplikasi *database*. *Application server* bertanggung jawab menyediakan metode yang mengimplementasikan *business logic* dari aplikasi *web* yang biasanya memiliki suatu bagian yang mengakses informasi dari *database*. Ketiga komponen tersebut, bisa dijalankan dalam satu mesin, atau setiap mesin untuk masing-masing komponen.

2.7. Rencana Pengujian

Dalam waktu satu dekade, Internet telah berkembang pesat dan akan terus tumbuh dalam rata-rata eksponensial. Sistem berbasis web dan aplikasi akan mengirim isi dan fungsi yang beragam ke berbagai jenis pengguna. Interaksi antara sistem *Web* dan sistem informasi *backend* juga menjadi lebih kompleks.

Walau pengembangan dan pengaturan *web* mengalami perkembangan yang pesat, namun hal ini dilakukan dengan dasar sesuai kebutuhan sehingga menghasilkan web dengan kualitas yang buruk. Tantangan-tantangan ini telah melahirkan sebuah disiplin ilmu yaitu *Web Engineering*. *Web engineering* adalah cara pembangunan dan pengorganisasi pengetahuan tentang pembangunan aplikasi *web*, penerapan pengetahuan tersebut untuk membangun *web* dan cara mengelola kompleksitas dan keberagaman aplikasi *web* [13].

Pada [13] dibahas pengujian dan evaluasi *web* diantaranya meliputi :

- *Browser compatibility* : aplikasi yang dibangun harus mempertimbangkan jenis aplikasi *browser* yang mengaksesnya termasuk dukungan format yang dapat ditangani.
- *Page display* : halaman dari aplikasi *web* yang ditampilkan harus dapat menyesuaikan dengan aplikasi *browser* yang digunakan.

- *Usability* : aplikasi yang dibuat harus dapat mewakili fungsi yang ditanganinya dan mudah digunakan.
- *Content analysis* : konten yang digunakan adalah yang tidak memiliki masalah. Misalnya masalah hak cipta, royalty, dan lain-lain.
- *Availabitlity* : pengguna dapat mengakses aplikasi *web* dari mana saja dan kapan saja.
- *System integration* : penggunaan format data yang memudahkan untuk berhubungan dengan aplikasi lain atau sistem yang lebih besar.
- *Performance* : waktu yang diperlukan untuk mengirimkan hasil kepada pengguna.
- *Security* : memperhatikan hal keamanan dan autentikasi pengguna dalam mengakses aplikasi

Dalam hal aplikasi khusu *mobile learning*, Parsons [14] secara spesifik menentukan aspek kualitas teknik yang harus dipenuhi oleh sebuah *m-learning* yaitu :

- Reliabilitas, menentukan tipe media yang disediakan.
- Ukuran layar dan resolusi, menentukan bagaimana sebuah halaman ditampilkan.
- *Standard tools* dan metada, untuk menjamin konsistensi isi.