

BAB 1

Pengenalan Pemrograman WEB

1.1 Mengapa harus dengan Web?

Selamat datang pada pelajaran tentang web programming. Untuk memulainya , dimulai dengan sebuah pengertian yang baik tentang bagaimana web dapat berguna untuk perusahaan dan programmer seperti pada pemrograman Web.

1.1.1 Lingkungan Teknologi Netral.

Pertama-tama, sebuah pemikiran yang baik tentang aplikasi pada internet yaitu 'Net adalah technology aman lingkungan. Komunikasi dengan bermacam aplikasi pada web yang dijalankan melalui popular protocol(HTML/HTTP) hal itu tidak dibutuhkan user untuk mempunyai operation system yang khusus maupun klien yang diprogramkan pada bahasa pemrograman tertentu atau framework. Semua user ingin menggunakan web browser, aplikasi standart ini terdapat pada operation system apapun.

Dikarenakan program yang dibutuhkan pada pembelajaran ini hanyalah sebuah web browser, tidak perlu membagi program-program melalui CD. User tidak perlu juga melalui sebuah proses instalasi yang panjang; yang akan mereka perlukan adalah lokasi aplikasi di Internet, dan mereka telah siap.

Manfaat yang lain yang dimiliki biner dari suatu program yang terdapat pada server yang diakses terdapat pada user computer yaitu permasalahan yang umum yang terkait dengan update program, seperti kebutuhan pada waktu tertentu melihat kemungkinan versi terbaru dari suatu program, permasalahannya adalah bagaimana cara mendapatkan program updating; disisihkan secara bersamaan, user tidak perlu diberitahu atas program yang sudah terupdate; semua yang dibutuhkan untuk mengupdate program pada web server dan secara otomatis semua user akan menggunakannya setelah itu akan menikmati manfaat dari update.

1.1.2 Arsitektur Client Server

1.1.2.1 Thick dan thin clients

Aplikasi web adalah jenis aplikasi yang menggunakan *arsitektur client-server*. Pada jenis arsitektur ini, sebuah program *client* terhubung pada sebuah *server* untuk informasi yang dibutuhkan untuk melengkapi tugas-tugas yang telah diset oleh user. Ada yang disebut *thin client* (client tipis), dan ada juga *thick client* (client tebal).

Thin client adalah clients yang hanya berisikan sedikit dari apa yang diperlukan untuk pengalaman user, kebanyakan hanya interface. Semua logika bisnis, semua data, terkecuali yang disediakan oleh user, berada di dalam server. Thick clients adalah clients yang sama, kecuali pada interface, juga berisi beberapa, jika tidak banyak, logika pengolahan diperlukan untuk tugas-tugas user yang spesifik.

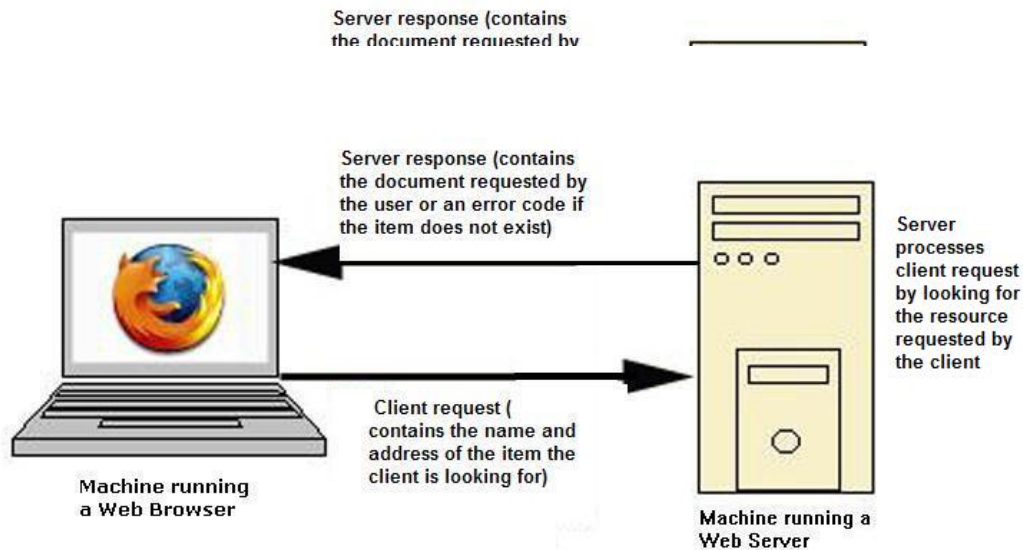
1.1.2.2 Arsitektur Client-Server dari perspektif Web

Dari definisi di atas, kita dapat menyimpulkan bahwa client digunakan untuk aplikasi web thin clients. Program client, pada hal ini adalah browser, hanya sebuah interface yang oleh user digunakan untuk melaksanakan tugas-tugas. Yang lainnya, dari data yang user perlukan untuk dioperasikan, logika yang menentukan aliran program dan eksekusi, berada pada server.

Dari suatu perspektif web base disini adalah tugas-tugas dari server client:

Web server

Gambar 1.1 : Tanggung Jawab Server



Pada dasarnya, server menerima permintaan-permintaan dari para client web browser dan kemudian meresponnya. Beberapa permintaan yang datang dari client disertai nama dan alamat item yang client cari, sebagaimana beberapa data user yang disediakan. Server menerima permintaan tersebut, memprosesnya, dan kemudian merespon data yang dicari oleh client atau sebuah kode error yang mengindikasikan bahwa item tidak terdapat pada server atau jika terjadi beberapa error lain.

Web client

Tugas browser adalah menyediakan user sebuah interface dimana akan meminta server dan menampilkan respon dari server.

Ketika user meminta server (sebagai contoh, mendapatkan dokumen, atau mungkin mengirim (*submit*) sebuah form), browserlah yang memformat permintaan tersebut ke dalam sesuatu yang server dapat mengerti. Begitu server telah selesai memproses permintaan dan kemudian mengirim respon, browser mengambil data yang diperlukan dari respon yang diberikan server dan kemudian merendernya untuk ditampilkan ke user.

HTML

Bagaimana cara browser mengetahui apa yang harus ditampilkan ke user? Sebagian besar situs web tidak hanya berisi teks sederhana, tetapi disertai grafis atau memiliki form yang dapat memanggil suatu data. Bagaimana masing-masing browser mengetahui apa yang harus ditampilkan?

Jawabannya adalah HTML, sebuah singkatan dari **H**ypertext **M**arkup **L**anguage. HTML dapat dimengerti sebagai sebuah kumpulan perintah-perintah untuk web browser tentang bagaimana menampilkan isi ke user. Itu merupakan standar terbuka yang telah di update oleh W3C atau *World Wide Web Consortium*.

Karena merupakan sebuah standar terbuka, setiap orang mengaksesnya, berarti bahwa browsers dikembangkan dengan standar itu di pikiran. Lebih lanjut berarti bahwa semua browsers mengetahui apa yang dilakukan ketika itu memecahkan HTML, meskipun beberapa browsers yang lebih lama mungkin memiliki permasalahan pada perenderan beberapa halaman yang ditulis menggunakan versi HTML yang lebih baru yang telah diupdate setelah pengembangannya.

HTTP

Definisi

HTTP singkatan dari "*HyperText Transfer Protocol*". Merupakan sebuah protokol jaringan dengan fitur-fitur Web-specific yang berjalan pada bagian teratas dari dua lapisan protokol lain, TCP dan IP. TCP adalah sebuah protokol yang bertanggung jawab memastikan file telah dikirim dari akhir network telah lengkap dikirimkan, berhasil pada tujuannya. IP merupakan sebuah protokol yang mengarahkan (*routing*) file dari satu host ke host lain pada jalannya untuk tujuan. HTTP menggunakan dua protokol ini untuk memastikan bahwa permintaan dan respon telah lengkap dikirimkan diantara masing-masing akhir komunikasi.

HTTP menggunakan urutan Request/Response: Sebuah *HTTP client* membuka koneksi dan mengirim sebuah *pesan permintaan* pada *HTTP server*; server kemudian mengirimkan *pesan respon*, biasanya berisikan resource yang diminta. Setelah mengirimkan respon, server menutup koneksi (membuat HTTP menjadi protokol tanpa status, contoh, tidak memelihara beberapa informasi koneksi diantara transaksi).

Format dari pesan permintaan dan respon adalah sama, dan berorientasikan bahasa inggris. Kedua jenis pesan mengandung :

- Sebuah garis inisial
- Nol atau lebih garis header
- Sebuah garis kosong(i.e sebuah CRLF oleh dirinya sendiri), dan pesan body optional (e.g. sebuah file, atau data query, atau keluaran query).

HTTP Requests

Permintaan-permintaan dari client ke server berisikan informasi tentang macam-macam data yang user inginkan. Salah satu item informasi yang dienkapsulasi pada permintaan HTTP adalah sebuah *nama method*. Ini memberitahu server macam-macam permintaan yang dibuat, sebagaimana sisa pesan dari client diformat. Ada dua protokol yang mungkin akan Anda gunakan : *GET* dan *POST*.

GET

GET adalah method HTTP paling sederhana dan digunakan sebagian besar untuk meminta resource tertentu dari server, apakah berupa halaman web, file gambar grafis, atau sebuah dokumen, dan lain-lain.

GET dapat juga digunakan untuk mengirim data di atas server, meskipun demikian hal itu mempunyai batasan-batasan. Jumlah total karakter yang dapat dienkapsulasi ke dalam permintaan *GET* adalah terbatas, sehingga untuk situasi dimana banyak data perlu dikirimkan ke server, tidak semua pesan dapat disampaikan.

Batasan lain method *permintaan GET* ketika mengirim data adalah data yang Anda kirim menggunakan method ini *ditambahkan* pada URL yang Anda kirim ke server. (Untuk sekarang, asumsikan URL sebagai alamat unik yang akan Anda kirim ke server sebagai penandaan lokasi yang Anda minta). Salah satu permasalahannya adalah URL dari beberapa permintaan yang Anda inginkan ditampilkan pada *bar* browser pada beberapa browser. Hal ini berarti, bahwa beberapa data sensitif seperti password atau informasi kontak (*contact information*) dapat terlihat oleh siapapun.

Keuntungan dari penggunaan *GET* dalam pengiriman data di atas server adalah permintaan URL dari permintaan *GET* dapat dibookmark oleh browser. Hal ini berarti bahwa user dapat dengan mudah membookmark permintaannya dan mengakses setiap saat dari pada melalui proses tiap waktu. Hal ini juga dapat membahayakan; jika bookmark secara fungsional bukan merupakan sesuatu yang Anda inginkan pada user Anda, sebagai gantinya menggunakan method lain.

Di bawah ini merupakan URL yang dihasilkan oleh permintaan *GET* :

~~http://www.konrad.com/servlet/ServletURL?URL=2355&Host=~~

Semua item sebelum tanda tanya (?) merupakan URL asli permintaan (dalam hal ini ~~http://www.konrad.com/servlet/ServletURL~~). Setelah itu, berikutnya adalah *parameters* atau data yang Anda kirim ke server.

Mari kita lihat secara seksama bagian tersebut. Berikut ini parameter yang ditambahkan pada permintaan :

~~newsItem=2359&filter=true~~

Pada permintaan *GET*, parameters disandikan sebagai nama dan nilai. Anda tidak mengirim nilai data ke server tanpa mengetahui secara spesifik untuk apakah nilai tersebut. Nama dan nilai disandikan sebagai berikut :

name=value

Dan juga, jika terdapat lebih dari satu kumpulan parameter, akan dipisahkan menggunakan tanda ampersand (&). Sehingga , dalam hal ini, nama-nama parameter yang kita spesifikkan ke server adalah *newsItemID* dan *filter*, dengan nilai *2359* dan *true*, berturut-turut.

POST

Jenis lain dari method permintaan yang pasti akan digunakan adalah permintaan *POST*. Jenis permintaan ini didesain seperti browser dapat membuat permintaan kompleks dari server. Mereka didesain sehingga user, melalui browser, dapat mengirim banyak data ke server. Form kompleks secara umum dicapai dengan menggunakan permintaan *POST*, sebagaimana form sederhana yang memerlukan proses upload file ke server.

Satu perbedaan yang nyata antara method *GET* dan *POST* terletak pada cara mengirimkan data ke server. Seperti yang dinyatakan sebelumnya, *GET* hanya menambahkan data ke URL yang akan mengirim. *POST*, di sisi lain, mengenkapsulasi atau menyembunyikan data di dalam body pesan (message body) yang dikirim. Ketika server menerima permintaan dan menentukan bahwa itu merupakan sebuah permintaan *POST*, dapat dilihat dari body pesan data tersebut.

HTTP Response

HTTP merespon dari server yang berisi headers dan body pesan, seperti yang permintaan HTTP lakukan. Mereka menggunakan kumpulan header yang berbeda, meskipun demikian disini kita tidak perlu terlalu dalam membahasnya secara detail. Cukup dengan mengatakan bahwa headers berisi informasi tentang protokol HTTP yang digunakan pada server, sebagaimana tipe dari isi yang dienkapsulasi ke dalam body pesan. Nilai dari tipe isi adalah *MIME-type*. Ini akan memberitahu browser jika pesan berisi HTML, gambar, atau tipe lainnya.

Dynamic over Static pages

Macam-macam content/isi yang dapat dilayani oleh web server dapat berupa *statis* atau *dinamis*. Content statis adalah isi yang statis atau tidak dapat dirubah. Content jenis ini biasanya hanya berada pada storage/penyimpanan dimana server dapat mengaksesnya dan akan diambil berdasarkan permintaan. Ketika mereka dikirim sebagai respon dari server, cara mereka dikirim sama persis seperti ketika mereka berada pada server.

Contoh dari content statis meliputi kumpulan artikel surat kabar, gambar keluarga dari galeri foto online, atau bahkan mungkin salinan online dokumen ini!

Dinamic content, pada sisi lain, berubah menurut input dari user. aplikasi apa pada server yang dapat mengakses pada tipe content ini yaitu semacam template yang mereka dapat ketahui yang mengacu pada bagaimana dokumen dapat dikirim dan akan terlihat secara umum, template ini kemudian membuat persetujuan pada parameter yang dikirim oleh user dan dikembalikan pada klien.

Hal itu dapat dikatakan sebagai berikut, halaman dinamis mempunyai lebih banyak fleksibilitas dan kegunaan dari pada halaman statis. Disini terdapat beberapa pasangan skenario dimana hanya content dinamik yang akan cocok:

- **Halaman web hanya didasarkan pada data yang diberikan oleh user.** Sebagai contoh, halaman dari hasil mesin pencari dihasilkan oleh cara ini, dan program yang memproses pesanan untuk lokasi e-commerce melakukannya dengan baik.
- **Data sering berubah,** sebuah laporan-cuaca atau halaman berita utama akan membangun halaman yang dinamik, mungkin menampilkan kembali halaman yang dibangun sebelumnya jika masih tergolong sebagai berita terbaru.
- Halaman web menggunakan informasi dari database perusahaan atau sumber yang lain.

Penting untuk disadari, web server dengan sendirinya tidak mempunyai kemampuan untuk melayani content dinamic, web server membutuhkan aplikasi yang dapat mereka akses untuk membangun content yang dinamic. Termasuk memerlukan aplikasi tersendiri untuk membuat content dinamic, web server selalu membutuhkan aplikasi tersendiri yang akan menyimpan informasi user yang bersangkutan (seperti mengumpulkan berdasarkan format) kedalam media penyimpanan, Anda tidak dapat mengharapkan untuk membuat sebuah form, memerintahkan user untuk memasukkan data kedalamnya, mengirimnya ke server, dan server secara otomatis mengetahui apa yang akan dilakukan pada data tersebut.

Kita kini berada dalam bagian diskusi kita dimana kita dapat dengan jelas menunjuk bahwa pembuatan dari aplikasi web ini yang merupakan pokok pembahasan pada bab ini, jadi, bagaimana cara yang kita lakukan untuk membuat aplikasi ini?

Pada bab ini, kita akan mempelajari dasar pada teknologi berbasis Java untuk membuat aplikasi web kita. Lebih spesifik lagi, kita akan membuat penggunaan APIs secara ekstensif yang disediakan pada web tier dalam spesifikasi J2EE (Java 2 Enterprise Edition).

1.1.3 Pengenalan J2EE Web Tier

Platform Java 2 Enterprise Edition (J2EE) adalah suatu platform yang diperkenalkan untuk development perusahaan aplikasi dalam suatu component-base. Model aplikasi yang digunakan pada platform ini adalah model aplikasi *distributed multi-tier*. Aspek yang diberikan dari model ini hanya berarti kebanyakan rancangan aplikasi dan pengembangan dalam platform ini diharapkan dapat mempunyai component yang berbeda yang di dapat dijalankan pada mesin yang berbeda pula. Multi-tier memisahkan bagian aplikasi yang dirancang dengan berbagai bagian separasi dengan berbagai komponen utama aplikasi

tersebut. Suatu contoh aplikasi multi-tier adalah suatu aplikasi web : layer presentasi (client browser), layer bussines logic (program yang berada pada web server), dan layer penyimpanan (database yang akan menangani data aplikasi tersebut) terpisah dengan jelas, tetapi secara keseluruhan bertujuan untuk menciptakan sebuah aplikasi bagi user.

Salah satu strata dalam platform J2EE ketika sebelumnya berupa *web-tier*. Strata ini diuraikan sebagai layer yang saling berhubungan dengan browser dalam rangka menciptakan content yang dinamis. Ada dua teknologi pada layer ini : servlet dan `JavaServerPage`.

Gambar 1-3 : Web Tier pada platform J2EE (Gambar dari J2EE Tutorial)

1.1.3.1 Servlets

Teknologi Servlet adalah jawaban utama yang terdapat pada Java untuk menambahkan fungsi ke server yang digunakan untuk merespon permintaan dari model. Mereka mempunyai kemampuan untuk membaca kumpulan data yang diminta oleh server dan menghasilkan response yang dinamis yang berdasarkan pada data tersebut, servlet tidak terbatas pada kondisi HTTP; seperti dinyatakan sebelumnya, mereka diterapkan untuk scenario manapun yang menuntut request dari object model. Kondisi HTTP pada saat ini adalah yang digunakan pertama kali, jadi Java menyediakan versi spesifikasi HTTP yang mengimplementasi pada fitur-fitur spesifik HTTP.

1.1.3.2 JavaServerPages.

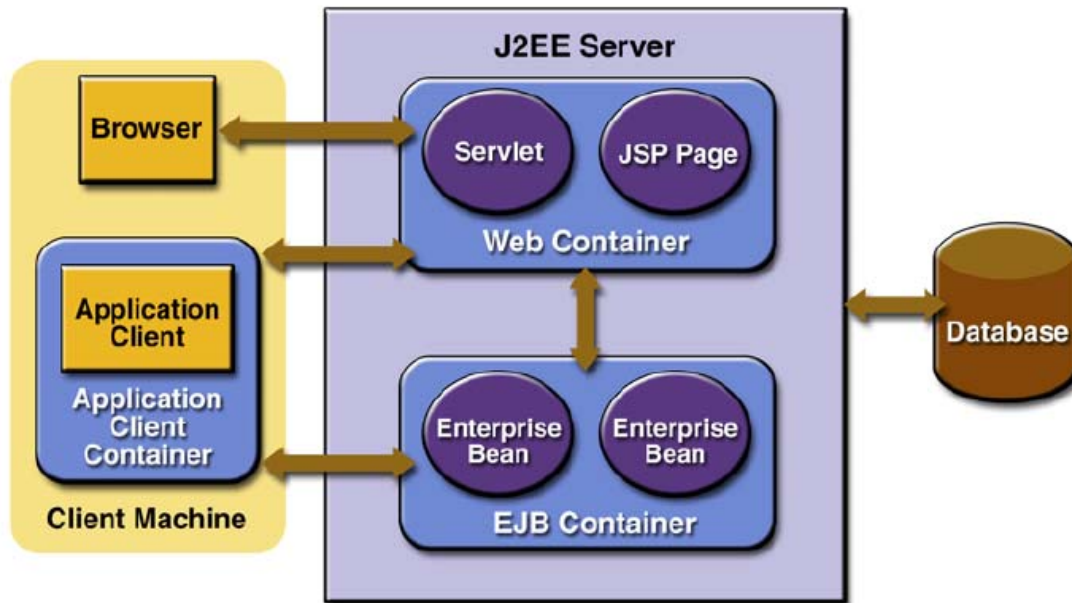
Salah satu dari kelemahan dalam penggunaan servlet yaitu pada proses generate sebuah respon dari klien yang berformat HTML akan dikirim kembali. Sejak servlet berupa class bahasa pemograman Java, mereka menghasilkan keluaran dengan cara lain pada pemograman Java seperti : mencetak karakter String ketika mengeluarkan output, dalam hal ini HTTP-response, bagaimanapun, HTML sangat complex dan sangat sulit untuk melakukan proses encode HTML melalui penggunaan String literal. Juga, melibatkan jasa suatu perancang gafis dan perancang halaman web untuk membantu didalam bagian halaman statis akan sulit atau mungkin mustahil : kita akan mengharapkan dia untuk mempunyai sedikit pengetahuan tentang Java.

Inilah dimana teknologi `JavaServerPage` masuk. JSP terlihat hanya seperti HTML. Hanya dapat mengakses kepada semua hal yang dynamic dari servlet dengan menggunakan script dan bahasa expression. Karena terlihat seperti HTML, designer berkonsentrasi pada desain sederhana HTML dan hanya meninggalkan sisa ruang kode untuk developer untuk mengisi dengan content dynamic.

1.1.3.3 Containers

Pusat dari konsep aplikasi J2EE adalah container, semua komponen J2EE, mencakup komponen web (servlet,JSPs) bersandar pada keberadaan suatu Container; tanpa container yang sesuai, mereka tidak akan dapat dijalankan.

Gambar 1-4 : Containers pada platform J2EE (Gambar dari J2EE Tutorial)



Barangkali cara untuk menjelaskan hal ini adalah untuk berpikir tentang ragam pelaksanaan program Java secara normal. Program Java, untuk dijalankan, harus memiliki cara utama yang menggambarkan ; menandai start ini pada pelaksanaan program dan menjalankan method ketika program diexecute dari command line.

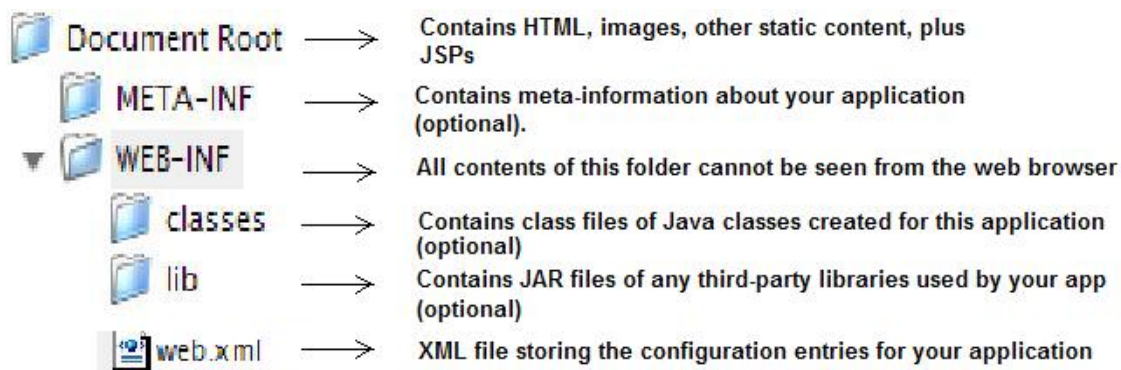
Tetapi, ketika kita dapat lihat kemudian, servlet tidak ditemukan cara utama. Atau jika ada sesuatu menemukan(desain program yang buruk), hal itu tidak menandai saat menjalankan eksekusi program,ketika seorang user membuat permintaan HTTP untuk sebuah servlet, metoda yang digunakan tidak dipanggil secara langsung. Sebagai gantinya, server tidak menyampaikan permintaan tadi kepada servlet, melainkan kepada container dimana servlet dikembangkan, kemudian container bertanggung jawab atas pemanggilan metode yang sesuai didalam servlet, tergantung pada jenis permintaan user.

Fitur yang diberikan oleh Container

- **Pendukung Komunikasi.** Container menangani semua kode yang penting dari servlet Anda untuk berkomunikasi dengan web server. Tanpa container, developer mungkin harus menuliskan kode yang akan menciptakan suatu koneksi dari server ke servlet (dan sebaliknya) dan mengatur bagaimana mereka bertemu satu sama lain pada tiap satuan waktu.
- **Management Lifecycle.** Container menangani segalanya dalam kehidupan pada servlet Anda, mulai loading class, instantiation dan inisialisasi, dan mengoleksi sampah.
- **Pendukung Multithreading.** Container mengatur tugas dari urutan waktu yang baru setiap kali suatu servlet dibuat, NOTE : container tidak bertanggung jawab dari servletmu.
- **Pendukung JSP.** Halaman JSP, dalam memperkerjakan, harus mengcompile dalam kode Java, mengcompilanya, dan memanggil metode yang sesuai dalam kode.

1.1.3.4 Struktur Dasar Dari Aplikasi Web

Supaya container dapat mengenali aplikasi sebagai aplikasi web yang sah, harus dibentuk struktur direktori yang spesifik:



Gambar 1-5 : Struktur Direktori dalam Java Web Application

Gambar 1-5 menunjukkan struktur direktori yang diperlukan oleh container untuk mengenali aplikasi Anda, beberapa hal mengenai struktur ini:

Pertama, top level Folder (salah satu yang mengisi aplikasi Anda) tidak harus diberi nama *Dokument Root*. Sebenarnya pemberian nama apapun tidak mempengaruhi aplikasi, namun disarankan untuk menamainya sesuai dengan nama aplikasi Anda. Penamaan sebagai Document Root hanya menggambarkan direktori ini bertindak sebagai root folder pada file atau dokumen pada aplikasi Anda.

Kedua, folder lain dapat dimuat pada struktur direktori ini, sebagai contoh, karena developer mengharapkan untuk mengorganisir content mereka, mereka dapat menciptakan suatu folder *images* dalam document root untuk semua file gambar, atau mungkin suatu direktori *config* pada folder WEB-INF untuk menjaga informasi tentang

konfigurasi. Selama mengikuti alur struktur yang ditentukan seperti diatas, container mengijinkan penambahan direktori.

Ketiga: Kapitalisasi pada folder WEB-INF adalah disengaja, Lowercaps pada class dan Library adalah disengaja juga, tidak mengikuti aturan kapitalisasi manapun pada folder ini akan mengakibatkan aplikasi Anda tidak dapat melihat isi dari folder ini.

Keempat: Semua content atau isi dari folder WEB-INF tidak dapat dilihat dari browser. Container secara otomatis mengatur berbagai hal, seperti : pada tampilan browser, folder tersebut tidak ada atau tidak diketahui. Mekanisme ini melindungi sumberdaya vital seperti pada file class Java, konfigurasi aplikasi, dan lain sebagainya. Content pada folder ini hanya dapat di akses oleh aplikasi Anda.

Kelima: HARUS ADA SUATU FILE dengan nama web.xml didalam folder WEB-INF. Meskipun, sebagai contoh, aplikasi web Anda hanya berisi content statis dan tidak menggunakan class Java atau file library, container masih akan mempersyaratkan bahwa aplikasi Anda menggunakan dua hal tersebut.