BAB II

LANDASAN TEORI

Bab ini bertujuan untuk memberikan latar belakang teori akan teknologi *Content Managements System*. Dengan ini diharapkan dengan ini pembaca dapat lebih memahami perkembangan proses pembangunan yang dijabarkan dalam penuisan tugas akhir ini.

2.1 Internet

2.1.1 Pengertian Internet

Istilah *internet* berasal dari bahasa Latin inter, yang berarti *"antara"*. Secara kata per kata *internet* berarti jaringan antara atau penghubung. Memang itulah fungsinya, *internet* menghubungkan berbagai jaringan yang tidak saling bergantung pada satu sama lain sedemikian rupa, sehingga mereka dapat berkomunikasi. Sistem apa yang digunakan pada masing-masing jaringan tidak menjadi masalah, apakah sistem DOS atau UNIX.

Secara harfiah, internet (kependekan dari kata inter-network) ialah rangkaian komputer yang berhubung menerus ke beberapa rangkaian. Internet ialah sistem komputer umum, yang berhubung secara global dan menggunakan TCP/IP sebagai protokol pertukaran paket (packet switching communication

protocol). Cara menghubungkan rangkaian dengan kaedah ini dinamakan internetworking.

Sementara jaringan lokal biasanya terdiri atas komputer sejenis (misalnya DOS atau UNIX), *internet* mengatasi perbedaan berbagai sistem operasi dengan menggunakan "bahasa" yang sama oleh semua jaringan dalam pengiriman data. Pada dasarnya inilah yang menyebabkan besarnya dimensi *internet*.

Internet merupakan hubungan antar berbagai jenis komputer dan jaringan di dunia yang berbeda sistem operasi maupun aplikasinya di mana hubungan tersebut memanfaatkan kemajuan media komunikasi (telepon dan satelit) yang menggunakan protokol standar dalam berkomunikasi yaitu protokol TCP/IP.

Menurut Turban (2005, p478), internet merupakan jaringan dalam jaringan yang menghubungkan komputer individual yang dimiliki oleh pemerintah, universitas, grup non-profit dann perusahaan. Interkoneksi ini dihubungkan dengan standar protokol yang bebas dan terbuka. Menurut Turban (2005, p50), internet adalah sistem jaringan komputer dan jaringan dari banyak jaringan yang meliputi seluruh dunia. Internet bersifat publik, kooperatif, dan mandiri yang memfasilitasi akses ke ratusan atau jutaan manusia di seluruh dunia.

Dengan demikian, definisi *internet* ialah "jaringannya jaringan", dengan menciptakan kemungkinan komunikasi antar jaringan di seluruh dunia tanpa bergantung kepada jenis komputernya.

2.1.2 Sejarah Internet

Internet dimulai pada tahun 1969. Pada waktu itu Departemen Pertahanan Amerika Serikat membangun sebuah jaringan komputer melalui proyek yang disebut ARPANET (*Advanced Research Projects Administrator Administration Network*) yang dimana mereka menjelaskan dan mendemonstrasikan bagaimana dengan hardware dan software komputer yang berbasis UNIX dapat melakukan komunikasi dalam jarak yang tidak terhingga melalui saluran telepon. Proyek ARPANET merancang bentuk jaringan, kehandalan, seberapa besar informasi yang dipindahkan dan akhirnya semua standar yang mereka tentukan menjadi asal mula pembangunan protokol baru yang sekarang dikenal sebagai TCP/IP (*Transmission Control Protocol/Internet Protocol*).

Pada awalnya, dibangunnya proyek tersebut adalah untuk kegiatan militer. Pada saat itu Departemen Pertahanan Militer Amerika Serikat merancang jaringan komputer yang tersebar dengan menghubungkan komputer di daerah-daerah vital untuk mengatasi masalah bila terjadi serangan nuklir dan untuk menghindari terjadinya informasi terpusat, yang apabila terjadi perang dapat muda dihancurkan.

Di tahun tersebut, jaringan itu hanya menghubungkan empat komputer yang tersebar di negara California dan Utah di Amerika Serikat, dan pada tahun 1969 mereka membentuk satu jaringan terpadu. Kemudian jaringan ini berkembang begitu cepat dan menghubungkan komputer-komputer yang berada di universitas, lembaga penelitian pemerintah dan instansi militer yang letaknya di

kota yang berbeda. Secara umum ARPANET diperkenalkan pada bulan Oktober 1972. Sejak awal, jaringan ini dirancang tetap berfungsi walaupun sebagian jalur rusak akibat perang.

Tidak lama kemudian proyek ini berkembang pesat di seluruh daerah, dan semua universitas di negara tersebut ingin bergabung, sehingga membuat ARPANET kesulitan untuk mengaturnya. Kemudian untuk solusinya, ARPANET dipecah menjadi dua bagian yaitu "MILNET" yang dilakukan untuk keperluan militer dan "ARPANET" baru yang lebih kecil untuk keperluan non-militer. Oleh sebab itu masing-masing komputer dapat berhubungan dengan komputer lain melalui berbagai macam jalur alternatif. Gabungan kedua jaringan tersebut akhirnya dikenal dengan nama DARPA internet, yang kemudian disederhanakan menjadi internet.

Pada tahun 1980-an, *National Science Foundation* membentuk NSFnet, yang menghubungkan setengah lusin super komputer pada kecepatan tertinggi waktu itu. NSFnet akhirnya mengambil alih ARPAnet.

a. *File Transfer Protocol* (FTP)

   FRP (*File Transfer Protocol*) adalah suatu protokol khusus yang memungkinkan untuk mentransfer file dari suatu komputer ke komputer lainnya dalam suaru jaringan internet. Dengan FTP, dapat dilakukan transfer file data grafik, data audio, video, spreadsheet, dokumen, serta file-file dengan format yang bermacam-macam jenisnya.

b. *Terminal Emulator* (Telnet)

Telnet adalah suatu program pengguna internet untuk berinteraksi dengan komputer lain yang seakan-akan pengguna-pengguna fasilitas ini berkerja di komputer lain yang dihubungkannya.

c. *Information Browsing* (Gopher)

Gopher merupakan sarana pertama yang memudahkan pencarian informasi di internet. Gopher memberikan kemudahan untuk menyusun daftar-daftar sehingga mudah untuk mencaru suatu informasi yang diinginkan.

d. *Advanced Browsing* (WWW, Mosaic)

WWW (*World Wide Web*) adalah hypermedia pada internet. Dokumen-dokumen informasi pada WWW disertai dengan grafik berwarna dan mendukung sistem multimedia dan memiliki link ke informasi lainnya. Untuk memperoleh dokumen WWW, kita menggunakan aplikasi browse Mosaic, Netscape, Websurfer, dan lainnya.

2.1.3 Peranan Internet

Sebelumnya penyebaran informasi lebih banyak dilakukan melalui media cetak (surat kabar, majalah, buku, atau brosur). Proses ini dapat memakan waktu yang cukup lama, bahkan hingga berhari-hari. Belum lagi ditambah dengan waktu pengiriman yang sangat bergantung pada lokasi yang dituju.

Kehadiran web menjadikan internet mengalami banyak sekali perubahan. Dengan menggunakan format HTML (*Hypertext Markup Language*), sebuah dokumen dapat memuat teks dan gambar sekaligus. Bahkan dengan perkembangan HTML, dokumen dalam bentuk multimedia (perpaduan grafik, animasi, suara, dan video) mampu ditampilkan.

Internet tidak hanya tempat orang berkomunikasi, tetapi juga menjadi tempat orang mendapatkan berita, informasi, dan pengetahuan. Di negara yang maju, ada yang menggunakan internet sebagai sambungan untuk video. Dengan demikian, informasi yang tersedia di internet menjadi daya tarik semua orang.

2.2 *Content Management System* (CMS)

2.2.1 Pengertian Sistem

Pengertian dan definisi sistem pada berbagai bidang berbeda-beda, tetapi meskipun istilah sistem yang digunakan bervariasi,semua sistem pada bidang-bidang tersebut mempunyai beberapa persyaratan umum, yaitu sistem harus mempunyai elemen, lingkungan, interaksi antar elemen, interaksi antara elemen dengan lingkungannya, dan yang terpenting adalah sistem harus mempunyai tujuan yang akan dicapai.

Berdasarkan persyaratan ini, sistem dapat didefinisikan sebagai seperangkat elemen yang digabungkan satu dengan lainnya untuk suatu tujuan bersama. Kumpulan elemen terdiri dari manusia, mesin, prosedur, dokumen, data

atau elemen lain yang terorganisir dari elemen-elemen tersebut. Elemen sistem disamping berhubungan satu sama lain, juga berhubungan dengan lingkungannya untuk mencapai tujuan yang telah ditentukan sebelumnya.

Terdapat beberapa definisi sistem yaitu :

Gordon B. Davis (1984) “Sebuah sistem terdiri dari bagian-bagian saling berkaitan yang beroperasi bersama untuk mencapai beberapa sasaran atau maksud“.

Raymond Mcleod (2001) “Sistem adalah himpunan dari unsur-unsur yang saling berkaitan sehingga membentuk suatu kesatuan yang utuh dan terpadu“.

2.2.2 Pengertian Konten

Konten (*Content*) pada intinya, tipe atau unit apapun dari informasi *digital*. Bisa berupa teks, gambar, grafik, *video*, suara, dokumen, catatan dan sebagainya. Atau dengan kata lain apasaja yang kemungkinan bisa di kelola dalam format elektronik.

2.2.3 Pengertian Manajemen Konten

Manajemen Konten (*Content Management*) adalah pengelolaan secara efektif dari konten yang dijelaskan diatas, dengan menggabungkan aturan-aturan, proses dan atau aliran kerja dalam suatu bentuk dimana penyimpanan elektroniknya lebih dipertimbangkan untuk dikelola daripada tidak dikelola.

2.2.4 Pengertian *Content Management System* (CMS)

Content Management System atau lebih populer dengan singkatan CMS, secara ringkas atau singkatnya dapat disebut sebagai sebuah sistem yang mengelola konten atau wacana. Pertama kali muncul sebagai jawaban atas solusi dari kebutuhan manusia akan penyediaan informasi yang sangat cepat

Hanya apabila konten yang akan dikelola tersebut diketahui maka dapat dilakukan penyaringan berdasarkan pilihan yang tersedia.

Sistem sendiri dapat dideskripsikan sebagai sebuah alat atau gabungan atau peralatan yang memfasilitasi produksi yang efisien dan efektif dari keluaran yang diinginkan menggunakan manjemen konten.

a. CMS adalah sebuah alat yang memungkinkan suatu keragaman teknik dan staf non-teknik untuk mebuat, merubah, mengelola dan akhirnya menerbitkan beragam konten (seperti teks, gambar, grafik, *video*, suara, dokumen dsb), sementara itu dibatasi dengan aturan-aturan, proses dan alur kerja yang terpusat memastikan konten tersebut benar.
b. CMS adalah sebuah sistem yang memberikan kemudahan kepada para pengguna dalam mengelola dan mengadakan perubahan isi sebuah *website* dinamis tanpa sebelumnya dibekali pengetahuan tentang hal-hal yang bersifat teknis. Dengan demikian setiap orang, penulis, maupun editor, setiap saat dapat menggunakannya secara leluasa untuk membuat, menghapus atau bahkan memperbaharui isi *website* tanpa campur tangan langsung dari pihak *webmaster* [4].

c. CMS dapat diartikan sebuah sistem yang digunakan untuk mengelola isi sebuah *web* secara dinamis [6].

d. CMS dapat diartikan sebuah teknologi baru yang masih asing bagi sebagian orang yang dapat membawa kesuksesan untuk setiap *web* [1].

Berdasarkan beberapa definisi di atas, secara sederhana dapat dikatakan CMS adalah suatu sistem yang dibuat dengan menggunakan teknologi, dalam hal ini skrip berbasis *server*, *database server* dan *webserver* oleh seorang yang mempunyai kemampuan memprogram *web* yang sangat baik agar pengelolaan *website* itu dapat dilakukan oleh siapa saja sehingga *content web* dapat selalu dinamis.

2.2.5 Karakteristik CMS

1. Fungsi CMS

Sebuah sistem CMS bisa dikatakan baik apabila telah memenuhi beberapa persyaratan yang meliputi beberapa fungsi pokok yaitu [4]:

a. Fungsi pembuatan isi (*content creation*)

Dalam bagian ini fungsi dibuat agar diarahkan untuk mampu menangani pengisian informasi secara mudah dan efisien, dimana seorang penulis informasi tidak perlu menggunakan tag HTML dalam membuat sebuah halaman *web*.

b. Fungsi manajemen isi (*content management*)

Dalam bagian ini fungsi dibuat untuk mengatur pusat isian, baik penampilan data yang sudah atau belum terpakai sehingga dalam proses penampilan informasinya menjadi mudah.

c. Fungsi menerbitkan isi untuk ditampilkan (*publishing*)

Dalam bagian ini fungsi dibuat untuk memuat fleksibilitas dan pengembangan halaman *web* yang dihasilkan. tampilan (*Layout*) halaman dispesifikasikan melalui sebuah halaman dengan pola yang telah disediakan (*template*)

d. Fungsi penampil isi (*presentation*)

Dalam bagian ini fungsi dibuat untuk menampilkan isi terhadap informasi yang telah tersimpan untuk ditampilkan di atas *browser*.

2. Manfaat CMS

Bila CMS dapat diterapkan dengan baik, ada beberapa manfaat dari penggunaan CMS yang dapat dijabarkan sebagai berikut:

1. Manajemen data
2. Mengatur siklus hidup *website*
3. Mendukung *web templating* dan standarisasi
4. Personalisasi *website*
5. Sindikasi; *Sindikasi* memberikan kemungkinan kepada sebuah *website* untuk membagi isinya kepada *website-website* yang lain.
6. Akuntabilitas dan efisiensi

3. Prinsip CMS

CMS secara prinsip dapat digunakan untuk berbagai macam keperluan dan dalam berbagai kondisi, terutama yang berhubungan dengan dunia *website*, beberapa diantaranya yaitu:

a. Mengelola *website* pribadi.
b. Mengelola *website* perusahaan/bisnis.
c. Portal atau *website* komunitas.
d. Galeri foto, dan lain sebagainya.
e. Forum.
f. Aplikasi *E-Commerce*.

4. Penggolongan CMS

Karena disadari akan pentingnya CMS, banyak para pakar membuat aplikasi CMS dengan berbagai fitur handal yang ditawarkan, hal ini dapat di lihat dengan maraknya topik mengenai CMS di internet. Beberapa situs *web* yang mengkaji mengenai CMS di internet yaitu www.kyantonius.com, www.oscom.org, www.duniakita.net, dan www.steptwo.com. Oleh karena itu berdasarkan aplikasinya, CMS dibagi menjadi 2 golongan yaitu:

a. CMS Komersil

   Dibuat dan dikembangkan oleh perusahaan-perusahaan *software* yang menjalankan usahanya dengan motif mencari keuntungan. Beberapa

aplikasi CMS yang ada di pasaran yang cukup terkenal yaitu STELLENT (*Stellent Content Management Suite),* VIGNETTE *(V7 Content Management Suite),* INTERWOVEN *(Team Site 6.0.)*

b. CMS *Open source*

Dibuat dan dikembangkan oleh sekelompok orang atau perusahaan yang intinya memberikan sebuah alternatif murah dan terjangkau kepada para pengguna. Tersedia secara gratis dan dapat dipergunakan sesuai dengan kebutuhan tanpa ada batasan. CMS jenis ini juga memberikan akses kepada penggunanya untuk mengetahui kode-kode pemograman yang disertakan, sehingga memudahkan pengguna memodifikasi CMS di masa-masa yang akan datang. Kelebihan CMS jenis ini, yaitu kode pemrograman terbuka untuk umum, sehingga apabila ada *bugs*, para pengguna dapat saling bahu-membahu dalam hal melacak danmemperbaiki *bugs* yang ada, menambah dan meningkatkan fungsi dan kemampuan CMS dan memberikan dukungan teknis dan non-teknis kepada yang membutuhkan. Beberapa jenis aplikasi CMS *Open Source* yang ada dipasaran seperti ZOPE (*Content Management Framework),* OPEN CMS (*Open CMS*), POSTNUKE (*Post Nuke*), PHP Nuke, MAMBO *OPEN SOURCE* (*Mambo Open source*).

2.3 Object Oriented Software Engineering

2.3.1 Pengertian *Object Oriented*

Menurut Lethbridge dan Lagainere (2002, p29), paradigma *Object Oriented* dapat digunakan untuk memecahkan masalah semua dimana semua perhitungan dilakukan dalam konteks obyek (*instance* dari *class*).

2.3.2 Pengertian *Software Engineering*

Menurut Canadian Standard Association, *Software Engineeringi* merupakan suatu aktivitas sistematis yang terlibat dalam perancangan, implementasi dan pengetesan piranti lunak untuk mengoptimasi produk dan dukungannya.

Menurut Lethbridge dan Laganiere (2002, p5), *Software Engineering* adalah sebuah proses memecahkan masalah *customer* dengan pengembangan sistematis dan evolusi sistem piranti lunak yang besar dan berkualitas tinggi dalam biaya, waktu dan *constraint* lain.

2.3.3 Aktifitas Pengembangan dalam Object Oriented Software Engineering

a. *Requirement Elicitation*

Requirement Elicitation ialah suatu komunikasi antaa *developer*, klien dan *user*. Komunikasi tersebut digunakan untuk mendefinisikan suatu sistem baru. Kegagalan dalam komunikasi ini dapat menyebabkan sistem sulit digunakan atau gagal memenuhi *requirement user*. Hasil dari aktivitas ini adalah penjelasan sistem dalam bentuk *use case* dan *actor*.

Requirement elicitation dapat dibedakan pada aktivitas berikut ini:

1. Mengidentifikasi *actor*

   Dalam aktivitas ini, *developer* mengidentifikasi *user* yang berbeda yang akan didukung oleh sistem. *User* tersebut selanjutnya dikenal sebagai *actor*.



Gambar 2.1 *Actor* suatu sistem

2. Mengidentifikasi skenario

   Skenario merupakan penjelasan naratif dari apa yang dilakukan oleh *actor* dalam menggunakan aplikasi. Dalam aktivitas ini,

developer mengobservasi *user* dan mengembangkan seperangkat skenario rinci untuk fungsi umum yang disediakan oleh sistem.

3. Mengidentifikasi *use case*

   Setelah skenario disepakati *developer* dan *user*, *developer* menghasilkan seperangkat *use case* yang diturunkan dari perangkat skenario tersebut. *Use case* mewakii arus *event* secara lengkap dalam sistem. Dalam menjelaskan *use case*, *developer* menentukan batasan dari sistem yang akan dibuat.

4. Mengidentifikasi hubungan antar *use case*

   Dalam aktivitas ini, *developer* mengkonsolidasi *use case model* untuk menentukan hubungan antar *use case*, yaitu dapat berupa *communication, extend* dan *include*.

Tabel 2.1 Hubungan antar *use case*

Hubungan	**Fungsi**	**Notasi**
Association	Menggambarkan komunikasi antara *use case* dengan *actor* yang terlibat dalam *use case* tersebut.	———
Extend	Disebut pula *Extention*. Menggambar-kan sebuah *base use case* yang berhubungan dengan *optional* atau *extention use case* sebagai kondisi pada saat terjadi *exeptional*, yaitu pada saat kondisi	<<extend>> - - - - - -▶

	tidak terduga (kondisi abnormal), misalnya kesalahan melakukan suatu aksi oleh *actor*.	
Include	Disebut pula *Inclusion*. Menggambar-kan hubungan sebuah *base use case* yang menambahkan *behavior* ke *use case* lainnya, *base use case* memiliki urutan yang dibagikan oleh *inclusion use case*, sehingga pada *inclusion use case* tidak perlu deskripsi ulang mengenai *squence* yang sama dengan *use case* lainnya.	<<include>>
Generalization	Menggambarkan hubungan antara sebuah *use case* dengan *use case* lain yang lebih spesifik, dimana *use case* yang lebih spesifik menambah fitur ke *use case* yang lebih umum.	

b. *Analysis*

Pada tahap analisis, *developer* akan menghasilkan model sistem yang benar, lengkap, konsisten, tidak membingungkan, realistik, dan dapat

diverifikasi. *Developer* akan menemukan ambiguitas dan inkonsistensi dari *use case model* yang dihasilkan bersama klien. Hasil tahap ini dalah obyek model digambarkan pada *class diagram* dan *sequence diagram*.

Aktivitas yang terdapat dalam tahap analisis yaitu :

1. Mengidentifikasi obyek entitas

   Obyek entitas adalah obyek yang berpartisipasi dalam sebuah sistem, obyek tersebut meliputi *data source*, entitas yang perlu disimpan jalurnya.

2. Mengidentifikasi *attribute*

   Attribute adalah properti yang dimiliki oleh obyek secara individual. A*ttribute* dari *class* sama namun nilainya bisa berbeda untuk tiap obyek.

3. Mengidentifikasi *asscociation*

   Menggambar hubungan antara dua *class* atau lebih.

4. Memodelkan hubungan generalisasi antara obyek

   Generalization digunakan untuk mengurangi *redundancy* dari model analisis Jika dua atau lebih *class* menggunakan *attribute* atau *method* yang sama, maka persamaannya tersebut digabungkan menjadi sebuah *superclass.*

5. Memodelkan interaksi antar obyek dengan *sequence diagram*

   Sequence diagram menghubungkan *use case* dengan obyek. Diagram ini menunjukkan bagaimana perilaku sebuah *use case* atau skenario didistribusikan dalam obyek yang berpartisipasi.

6. Melakukan *review* terhadap model analisis

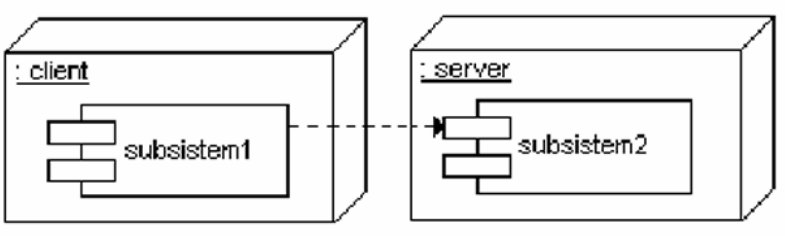
   Model analisis dibuat secara bertingkat dan berulang. Analisis model jarang yang sudah benar atau lengkap pada tahap awal. Beberapa perulangan analisis dengan klien dan *user* diperlukan sebelum model analisis dipertemukan dengan spesifikasi lengkap yang dapat digunakan oleh *developer* untuk memproses perancangan dan pengimplementasian.

c. *System Design*

Perancangan sistem juga merupakan proses transformasi model analisis menjadi model perancangan sistem. *System design* mencakup:

1. Memetakan subsistem ke dalam *processor* dan *components*

   Banyak sistem berjalan pada lebih dari satu komputer dan tergantung pada akses *intranet* atau *internet*. Penggunaan multi komputer membutuhkan koneksi terdistribusi yang memiliki kinerja tinggi. Untuk itu diperlukan alokasi subsistem ke komputer serta merancang infrastruktur yang mendukung komunikasi antara subsistem. Pemetaan subsistem ke *processor* dan *component* digambarkan dengan *deployment diagram.*



Gambar 2.2 Pemetaan Subsistem dengan Deployment Diagram

2. Mendefinisikan penyimpanan data

   Aktivitas ini adalah aktivitas yang berhubungan dengan peyimpanan data permanen untuk sistem secara umum.

   Secara umum, langkah pertama adalah menentukan obyek apa yang akan disimpan, kemudian bagaimana dan dimana data disimpan.

   Tipe penyimpanan data:

   a. *Flat File*

   b. *Relational-Database*

   c. *Object-Oriented Database*

3. Mendefinisikan kontrol akses

   Dalam sistem *multi user*, *actor* yang berbeda memiliki akses terhadap fungsionalitas dan data yang berbeda pula. Seorang *user* harus melakukan authentikasi ke dalam sistem, lalu sistem akan melakuakan enkripsi data untuk mencegah pengaksesan oleh *user* yang tidak berhak.

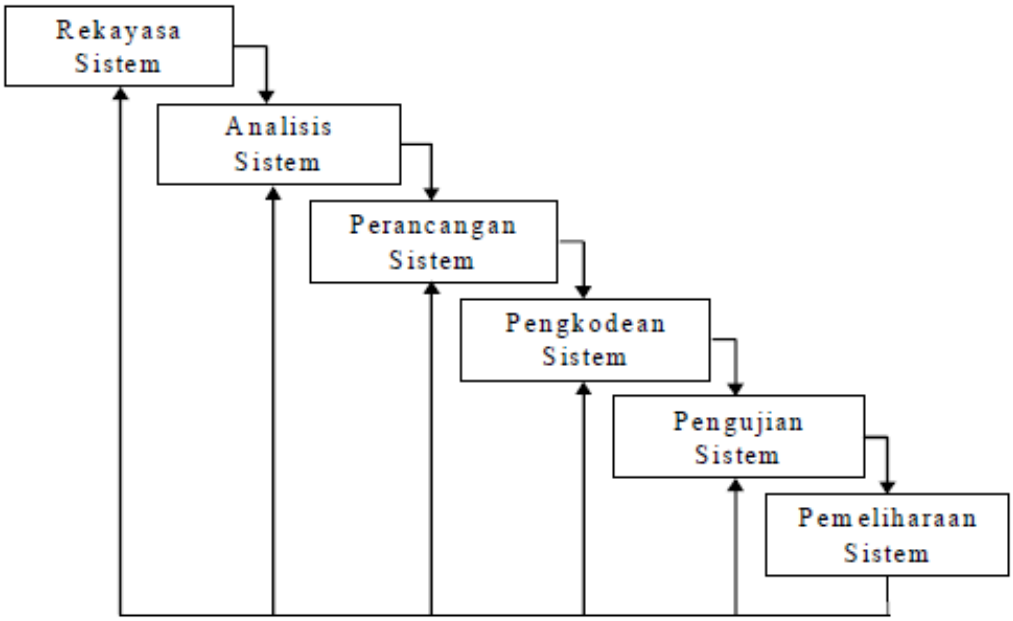
d. *Implementation*

   Pada tahap implementasi, *developer* menerjemahkan *object model* ke dalam *source code*, hal ini termasuk mengimplementasi *atributte* dan

method tiap obyek dan menyatukan setiap obyek sehingga bisa berfungsi sebagai sebuah sistem tunggal. Aktivitas implementasi menggabungkan celah antara perancangan obyek yang rinci dengan perangkat lengkap dari *file source code* yang dapat digabungkan menjadi satu.

2.4 Siklus Hidup Sistem (The Classic Life Cycle)

Siklus hidup sistem adalah proses evolusioner yang diikuti dalam menerapkan sistem atau subsistem informasi berbasis komputer. Terdiri dari serangkaian tugas yang erat yang mengikuti langkah-langkah pendektan sistem. Karena tugas-tugas tersebut mengikuti suatu pola yang teratur dan dilakukan secara top-down, sering juga disebut sebagai pendekatan air terjun (*waterfall approach*) bagi pengembangan dan penggunaan sistem.



Gambar 2.3 Model Waterfall

The *classic life cycle* merupakan pendekatan perkembangan perangkat lunak yang sistematik dan berurut, dimulai pada tingkatan dan kemajuan sistem pada seluruh analisis, perancangan, pengkodean, pengujian, dan pemeliharaan (Pressman, 1997, p37). Aktivitas-aktivitas dalam *the life cycle* ini melingkupi :

1. Rekayasa dan pemodelan sistem (*system engineering*)

   Oleh sebab perangkat lunak selalu merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat semua elemen sistem dan mengalokasikan beberapa *subset* dari kebutuhan ke perangkat lunak tersebut. Rekayasa dan pemodelan sistem menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis dan disain tingkat puncak.

2. Analisis kebutuhan perangkat lunak

   Pada aktivitas ini, proses pengumpulan kebutuhan diintensifikasi dan difokuskan, khususnya pada perangkat lunak. Kebutuhan baik sistem maupun perangkat lunak didokumentasikan.

3. Disain

   Disain perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur perangkat lunak, representasi *interface*, dan algoritma *procedural*. Proses desin menerjemahkan kebutuhan dalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode.

4. Pengkodean

   Disain harus diterjemahkan ke dalam bentuk bahasa mesin yang bisa dibaca. Langkah pengkodean dapat diselesaikan secara mekanis.

5. Pengujian

   Proses pengujian berfokus kepada logika internal perangkat lunak memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, mengarahkan pengujian untuk menemukan kesalahan dan memastikan input yang dibatasi akan memberikan hasil aktual, sesuai hasil yang dibutuhkan.

6. Pemeliharaan

   Perangkat lunak akan mengalami perubahan setelah disampaikan kepada pelanggan. Perubahan akan terjadi karena kesalahan-kesalahan ditentukan, karena perangkat lunak harus disesuaikan untuk mengakomodasi perubahanperubahan didalam lingkungan eksternal, atau karena pelanggan membutuhkan perkembangan fungsionalitas atau unjuk kerja. Pemeliharaan perangkat lunak mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

2.5 Delapan Aturan Emas Perancangan Antar Muka

Berikut ini merupakan prinsip-prinsip yang mendasari perancangan antar muka yang dapat diterapkan dalam banyak sistem (Shneiderman, p74,1998) :

1. Adanya konsistensi (*strive for consistency*.

   Aturan ini mungkin sulit dilakukan karena terdapat banyak bentuk dari konsistensi. Serangkaian konsistensi aksi dibutuhkan dalam situasi yang serupa, istilah yang sama dipergunakan dalam menu dan layar bantu, konsisten dalam warna, rancangan, huruf, dan lain-lainnya.

2. Pengguna yang sering menggunakan sistem dapat menggunakan *shortcut* (*enable frequent user to use shortcut*).

   Jika frekuensi pengguna suatu sistem meningkat, user ingin mengurangi jumlah interaksi. Singkatan, kunci yang spesial, perintah tersembunyi, dan fasilitas makro dihargai oleh pengguna yang sering menggunakan sistem.

3. Memberikan *feedback* yang berisi informasi (*offer information feedback)*.

   Setiap aksi pengguna harus mendapat feedback. Bagi aksi yang sering dan kecil, respon dapat berupa sesuatu yang sederhana. Akan tetapi, untuk aksi yang jarang dan utama, respon harus lebih penting.

4. Merancang dialog dengan penutupan (*design dialogs to yield closure*).

   Serangkaian aksi seharusnya diorganisasikan ke dalam kelompok: pemulaan, pertengahan, dan akhir. Feedback yang informative pada penyelesaian suatu kelompok aksi-aksi memberikan kepuasan penyelesaian bagi operator.

5. Menawarkan pencegahan kesalahan dan penanganan kesalahan sederhana (*offer error prevention and error handling*).

Perancangan suatu sistem sedapat mungkin membuat pengguna tidak dapat melakukan kesalahan serius. Jika pengguna melakukan kesalahan, sistem harus mendeteksi kesalahan tersebut dan memberikan instruksi yang mudah dan jelas untuk perbaikan.

6. Mengijinkan pembalikan aksi yang mudah (*permit easy reversal of actions*).

   Sedapat mungkin, pembalikan aksi dapat dilakukan. Fasilitas ini mengurangi kegelisahan pengguna karena pengguna tahu bahwa suatu kesalahan dapat dibatalkan, sehingga mendorong pengguna untuk melakukan penjelajahan pilihan-pilihan yang tidak biasa digunakan.

7. Mendukung tempat pengontrolan internal (*support internal locus of control*).

   Operator yang berpengalaman mempunyai perasaan yang kuat bahwa para operator tersebut bertanggungjawab terhadap suatu sistem dan sistem memberikan respon terhadap aksi-aksi yang dilakukan.

8. Mengurangi beban ingatan jangka pendek (*reduce short-term memory load*).

   Keterbatasan pemrosesan informasi manusia dalam ingatan jangka pendek memerlukan penayangan yang sederhana, menggabung penayangan banyak halaman dan waktu pelatihan yang cukup untuk kode dan rangkaian aksi.

2.6 Unified Modeling Language

Menurut Booch, Rumbaugh dan Jacobson (1998, p.3*), Unified Modeling Language* (UML) ialah bahasa pemodelan umum yang digunakan untuk melakukan spesifikasi, visualisasi, konstruksi dan dokumentasi artifak dari *software* sistem.

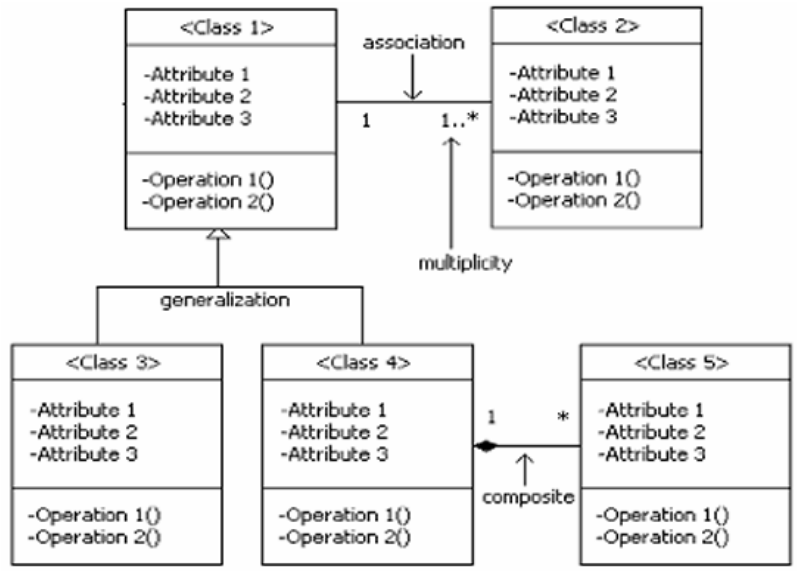
UML terdiri dari diagram, notasi, konsep dan aturan yang dipakai dalam memodelkan sistem. Diagram UML terdiri dari 9 jenis diagram yang memiliki fungsi dan notasi masing-masing. Diagram UML dibagi dalam 2 kategori yaitu :

1. Diagram yang menggambarkan struktur yang statis dari sistem:
   a. *Class Diagram* dan *Object Diagram*

      Menggambarkan struktur hubungan statis dari elemen-elemen sebuah model, diantaranya *class, package, dan relationship*. Pada *Object diagram* menggambarkan hubungan antar elemen dalam *model*, tapi dengan memakai obyeknya, bukan *class*nya. *Class* ialah kumpulan dari obyek yang memiliki *attribute, behaviour* atau *operation* yang sama.

      Class dan *object* dalam tahapan *design* digambarkan memiliki tiga bagian. Pada bagian atas diberi nama *class* atau obyek, bagian tengah merupakan *attribute* dan bagian bawah ialah *operation*. Dalam *class* dan *object diagram* terdapat beberapa istilah-istilah diantaranya yaitu :

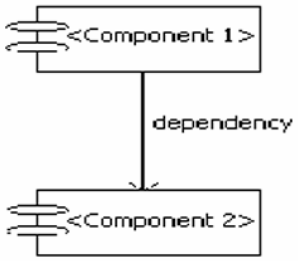
1. *Association Link* : mewakili hubungan antar dua obyek. *Association* adalah hubungan antar *class* dan mewakili kelompok *link.*
2. *Multiplicity* : banyaknya hubungan yang mungkin terjadi antar class.
3. *Aggregation* : bentuk khusus dari *association* yang menggambarkan bahwa satu *class* merupakan bagian dari *class* lainnya *"a part of"*. Dalam beberapa kasus, satu *class* dapat terbagi jadi beberapa *class*.
4. Generalisasi : hubungan antara *class* induk (*super class*) dengan *class* anak (*sub class*), dimana didalamnya terdapat hubungan *"is a"*. Pada hubungan ini *attribute* dan *behaviour* yang terdapat pada *super class* akan diwarisi oleh *sub class.*



Gambar 2.4 Class Diagram

b. *Component Diagram*

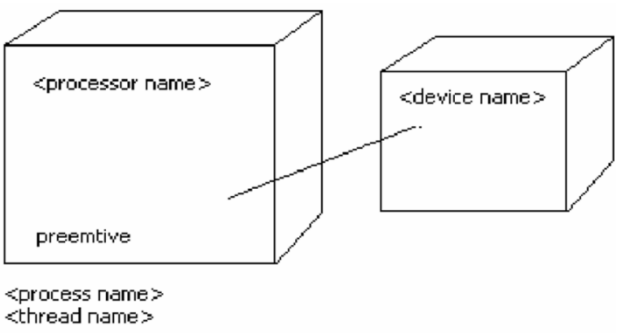
Merupakan gambaran aspek fisik sistem berbasis obyek dengan menunjukkan hubungan dan ketergantungan dalam serangkaian komponen fisik *software* termasuk *source code, run time (binary) code, executable file, table, library,* dan dokumen. Terdapat pula *interface, dependency, generalization, association, realization, notes, constraints, packages, subsystem* dari sebuah model. Diagram ini digunakan untuk memodelkan implementasi sistem yang statis.



Gambar 2.5 Component Diagram

c. *Deployment Diagram*

Menggambarkan sumber fisik dalam sistem, termasuk *node,* komponen dan koneksi (model implementasi sistem yang statistik). Dalam hal ini meliputi topologi *hardware* yang dipakai sistem.



Gambar 2.6 Deployment Diagram

2. Diagram untuk menggambarkan struktur yang dinamis dari sistem.

   a. *Use case Diagram*

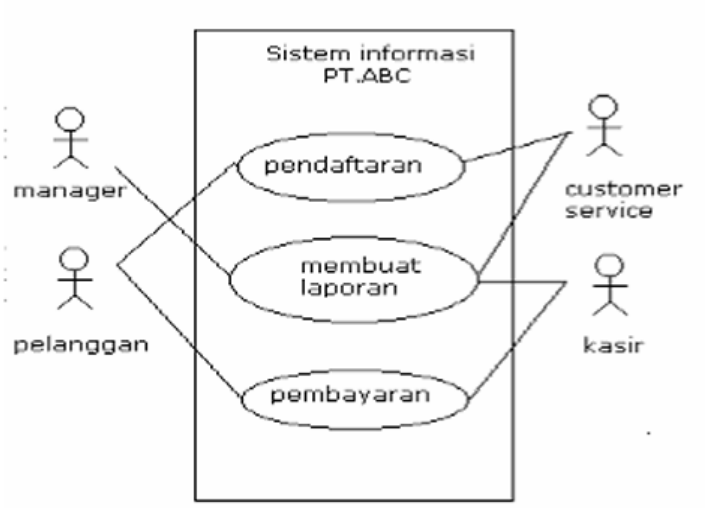
      Use case ialah salah satu model dalam analisis dan desain sistem berorientasi obyek. *Use case modeling* digunakan untuk mendokumentasikan *system behaviour* dan *subsystem* pada saat pengembangan sistem, termasuk di dalamnya fungsi internal suatu sistem (*use case*), pengguna sistem (*user*) dan hubungan interaksi antara keduanya (*use case diagram*).

      Use case diwujudkan dalam bentuk diagram dengan beberapa notasi baku yang ditujukan untuk memudahkan *developer* untuk melihat keseluruhan *behaviour* dari sebuah sistem; dan dalam bentuk teks, *use case* ini disebut dengan *narrative use case*, dimana proses yang ada dalam *use case* digambarkan dengan kata-kata sehingga menjadi lebih jelas.

      Bagian utama dalam *use case modelling*, yaitu :

      1. *Actor* : perwujudan dari pengguna sistem, proses dan segala sesuatu yang berinteraksi dalam sistem tersebut. *Actor* tidak termasuk dalam sistem, tapi dapat menggambarkan interaksi dari *external actor* dengan sistem tersebut. Setiap *actor* berinteraksi dengan satu atau lebih *use case* dengan pertukaran pesan.

2. *Use case :* merupakan bagian dari suatu sistem yang menyediakan sebuah fungsi atau tugas tertentu dan terdiri dari urutan aksi. *Use case* memperlihatkan *external behaviour* dari sebuah sistem yang dilihat dari segi pengguna eksternal. *Use case* tidak seperti *operation* karena sebuah *use case* dapat terus menerima *input* dari *actor* pada saat dijalankan.
3. *System Boundary :* menjelaskan batasan suatu sistem dengan lingkungannya, sehingga memberi batasan yang jelas sampai dimana suatu sistem bekerja, termasuk membatasi sistem dengan *actor* yang berada di luar dari sistem.

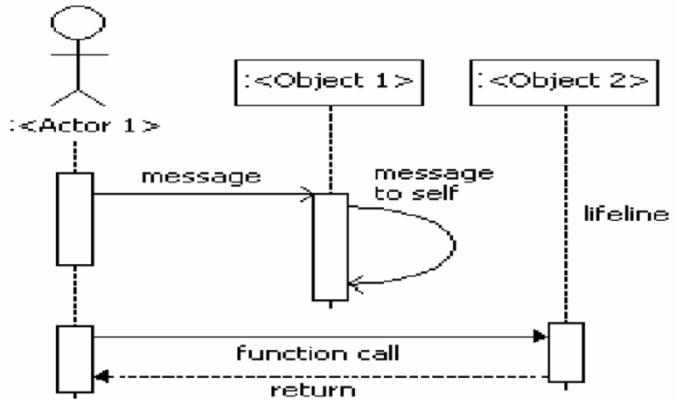


Gambar 2.7 Use Case Diagram

b. *Sequence Diagram*

Merupakan diagram yang menggambarkan pola hubungan diantara sekumpulan obyek yang saling mempengaruhi menurut urutan waktu. Sebuah obyek berinteraksi dengan obyek lain melalui

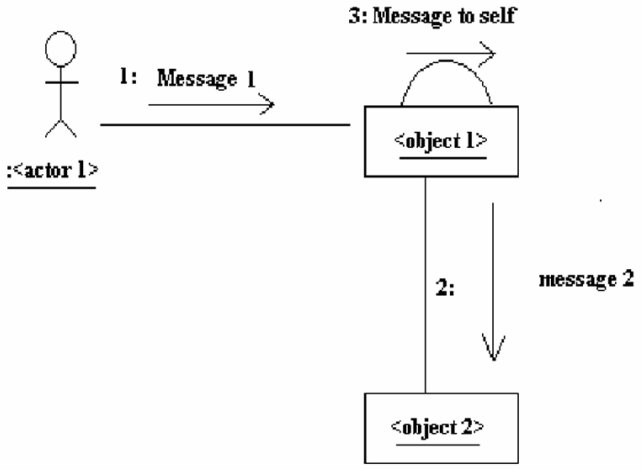
pengiriman *messages*. *Sequence diagram* ini biasanya digunakan untuk mengilustrasikan sebuah *use case*.



Gambar 2.8 Notasi Sequence Diagram

c. *Collaboration Diagram*

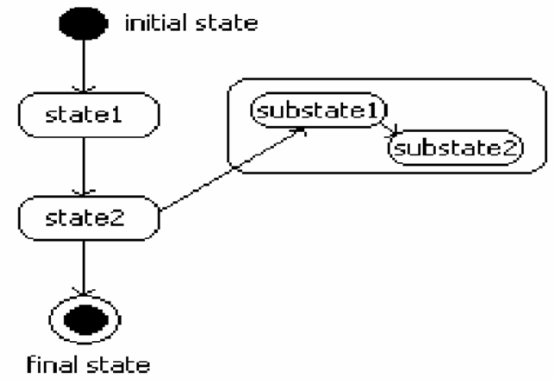
Collaboration diagram menggunakan interaksi antar obyek dengan serangkaian *message* dan juga menggambarkan struktur yang statis dan *behaviour* yang dinamis dari sistem.



Gambar 2.9 Notasi Collaboration Diagram

d. *Statechart Diagram*

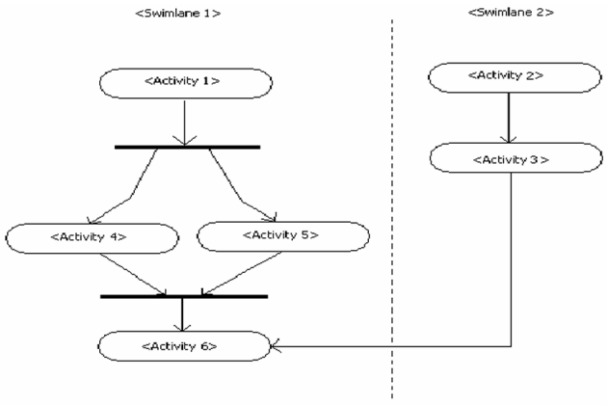
Merupakan diagram yang menggambarkan serangkaian *state* yang dapat terjadi pada sebuah *object*. *State* dapat berubah karena adanya suatu *event* atau kondisi tertentu. *State* merupakan kondisi dimana obyek melakukan sebuah kegiatan atau sedang menunggu event. *Statechart* diagram digunakan untuk menggambarkan *behaviour* yang penting dari subsistem atau sebuah obyek.



Gambar 2.10 Notasi Statechart Diagram

e. Activity Diagram

Activity diagram ialah diagram yang menggambarkan *flow of events* dari sebuah *use case* yang menjelaskan apa yang harus dilakukan oleh sistem agar dapat memberi nilai guna bagi *actor*.



Gambar 2.11 Notasi Activity Diagram

2.7 Pengembangan Aplikasi Berbasis *Framework*

PHP adalah sebuah bahasa pemrograman yang memungkinkan seorang developer (*programmer* atau *system analist*) membuat sebuah aplikasi berbasis web yang *powerful* sekaligus mampu mengampu database berskala besar. Dalam perkembangannya, seorang *programmer* PHP seringkali dituntut untuk menyelesaikan berbagai macam aplikasi dengan tingkat kerumitan yang cukup tinggi dalam waktu singkat. Di sisi lain, *programmer* juga dituntut untuk menciptakan sebuah dasar aplikasi yang bisa dikembangkan menjadi aplikasi lain dengan skala yang lebih besar dengan melibatkan banyak anggota tim.

Aplikasi web berskala besar seringkali diasosiasikan dengan indikasi-indikasi sebagai berikut :

a. Diakses oleh banyak orang (*public access*)
b. Melibatkan database dengan skala *record* diatas 1000
c. Mempunyai banyak modul, seperti modul berita, modul administrasi, modul keuangan, modul pencarian tingkat lanjut, *modul polling* dan sebagainya
d. Dikerjakan oleh sebuah tim pengembang dengan spesialisasi tugas

Berbicara tentang aplikasi web berskala besar tentu tidak bisa dilepaskan dari pembagian peran anggota tim. Aplikasi web, terutama web berskala besar, tidak hanya membutuhkan seorang programmer saja, akan tetapi melibatkan pula seorang *web desainer*, *system analist*, *database maintainer*, manajer keuangan,

manajer riset dan promosi dan manajer proyek yang akan mengatur jalannya pembuatan, pengembangan dan pemeliharaan aplikasi tersebut. Tingkat kerumitan dan kesamaan cara pandang inilah yang melahirkan konsep kerangka kerja (framework) dalam pengembangan aplikasi berbasis web. *Framework* memudahkan anggota tim untuk bekerja bersama dengan satu cara pandang, sekaligus memudahkan pembagian kerja. Pada umumnya, framework terdiri dari file-file pustaka berupa kelas-kelas PHP yang bisa digunakan dan dikolaborasikan dalam aplikasi yang sedang dikembangkan.

Di internet, tersedia puluhan, bahkan ratusan *framework* dan *Content Management System* (CMS). Sebut saja Mojavi, Prado, Agavi, Seagull, Code Igniter, Fusebox, PHP Nuke, Post Nuke, Mambo, Joomla, Wordpress, Blogspot, Aura, Endonesia, dan masih banyak lagi. Ada beberapa developer yang menyediakan *file* inti (*core*) *framework* saja, beserta manualnya, tanpa memberikan contoh-contoh aplikasi secara detail, misalnya Fusebox dan Mojavi (saat ini riset Mojavi terhenti, tapi dikembangkan menjadi framework baru, yakni Agavi). Ada juga *framework* yang benar-benar dibuat untuk membangun aplikasi *enterprise*, memberikan contoh yang detail, manual yang lengkap, namun memiliki ukuran besar dan memerlukan instalasi PHP 5 supaya bisa berjalan dengan baik, misalnya Prado. Terkadang orang mengartikan framework adalah dasar dari CMS. Di sisi lain, CMS tak lain juga sebuah *framework* dengan bermacam modul yang datang bersamanya.

2.8 Terminologi MVC

MVC singkatan dari *Model-View-Controller*. MVC adalah pola desain software yang membantu secara logis memisahkan kode program, membuatnya lebih *reusable*, *maintainable*, dan lebih baik secara keseluruhan.

***Model* :** representasi aktif dari struktur database. *Framework* berbasis MVC menekankan pentingnya disain *database* yang *valid*, termasuk hubungan antar tabel dalam database tersebut. *Model* memungkinkan *developer* untuk melakukan *query* antar *database* (bila memang diperintahkan oleh *controller*). Sebuah aplikasi MVC yang baik memisahkan antar tampilan dan logika kerja (*logic*). Model adalah sesuatu yang mewakili basis data yang diolah sedemikian rupa oleh *controller* (*logic*), untuk ditampilkan (*view*) hasil olahannya.

***View* :** tampilan luar dari aplikasi. *User* akan berhadapan langsung dengan tampilan ini. *View* tidak boleh mengandung *bussiness logic*, alias logika kerja aplikasi. Bisa dikatakan, view adalah template dari tampilan aplikasi yang memudahkan untuk menggantinya di kemudian hari.

Controller : inilah bagian yang paling signifikan dari sebuah aplikasi berbasis MVC. Benar dan tidaknya hasil olahan data akan sangat tergantung dari logika kerja aplikasi yang tersusun pada bagian controller ini. Controller akan mengolah data dari model, menyimpannya dalam variabel-variabel, dan menampilkannya pada view.

2.9 CakePHP

2.9.1 Profil

CakePHP adalah sebuah *framework open source* yang digunakan untuk mengembangkan aplikasi web dengan dasar kerja **CRUD (*Create, Read, Update, Delete*)**. CakePHP juga menjadi salah satu *framework* pilihan yang memungkinkan developer untuk membuat sebuah aplikasi web dengan karakter pengembangan **RAD (*Rapid Application Development*)**, yang memungkinkan untuk digunakan dan dikembangkan menjadi aplikasi lain yang lebih kompleks. CakePHP terdiri dari *file-file* pustaka (*library*), kelas-kelas, dan infrastruktur runtime yang terinspirasi oleh *framework* Ruby on Rails. CakePHP juga banyak digunakan oleh para *programmer* yang memilih untuk bekerja dengan struktur yang rapi dan padat tanpa kehilangan fleksibilitas pengembangan *framework*.

Pada tahun 2005, Michal Tatarynowicz mulai menulis beberapa kelas untuk sebuah dasar aplikasi RAD dengan menggunakan bahasa pemrograman PHP. Ia menyadari bahwa beberapa kelas yang ia ciptakan sangat memungkinkan untuk dikembangkan menjadi sebuah framework yang lebih lengkap dan praktis. Akhirnya, Michal mempublikasikan hasil kerjanya di bawah lisensi MIT Amerika Serikat, menamainya dengan Cake dan menawarkan pengembangannya pada komunitas developer PHP dan saat ini proyek tersebut dikenal dengan nama CakePHP.

CakePHP bersifat terbuka, bebas didapatkan, dan bebas dikembangkan. Komunitas pengembang CakePHP mengajak seluruh programmer PHP untuk bergabung dalam Cake Software Foundation Inc. yang membawahi proyek pengembangan CakePHP dan aplikasi berbasis CakePHP. Informasi tentang CakePHP bisa didapatkan pada situs resmi Cake Software Foundation di website www.cakephp.org.



Gambar 2.12 Logo Resmi CakePHP

Selain website resmi, para pengembang CakePHP juga biasa berdiskusi dengan menggunakan IRC (*Internet Relay Chat*). Di sini bisa dijumpai banyak ***Bakers***, sebutan untuk mereka yang menjadi developer CakePHP. Forum diskusi CakePHP ini bisa dikunjungi di channel #cakephp pada server irc.freenode.com.

2.9.2 Kelebihan

Selama ini, pemahaman yang kita peroleh tentang aplikasi web berbasis PHP adalah sebuah aplikasi yang terdiri dari file-file PHP yang berisi kode-kode dan tag html yang dicampur dengan kode-kode PHP atau lebih dikenal dengan nama spaghetti code.

Pengembangan aplikasi web dengan cara konvensional seperti ini tidaklah keliru, namun akan menjadi masalah besar apabila aplikasi ini membutuhkan banyak fitur dan pengembangan di kemudian hari. *Framework* CakePHP menjadi pilihan untuk mengatasi berbagai permasalahan yang muncul dalam pengembangan aplikasi web, karena beberapa kelebihannya, antara lain :

a. Open Source

   CakePHP bebas didapatkan dan dikembangkan. CakePHP mempunyai lisensi MIT. Ini adalah salah satu syarat yang baik untuk berkembangnya sebuah framework.

b. Riset yang terorganisir dengan baik

   Pengembangan framework bukanlah pekerjaan yang selesai dalam waktu 1-2 hari. Semakin terorganisirnya pengembangan sebuah framework akan berbanding lurus dengan kualitas framework yang dihasilkan. CakePHP memenuhi syarat ini dan dikembangkan dalam riset yang terorganisir dan berkesinambungan di bawah Cake Software Foundation.

c. Dokumentasi yang lengkap

   Framework yang baik harus mempunyai dokumentasi yang lengkap, karena sekumpulan kelas tanpa dokumentasi tak lebih dari sebongkah file yang asing dan menambah masalah developer. Untuk referensi lebih lengkap (gunakan manual dan API CakePHP), silahkan buka http://www.cakephp.org.

d. Kompatibel dengan PHP 4 dan PHP 5

CakePHP berjalan mulus di server Apache yang menggunakan PHP 4 maupun PHP 5. Fleksibilitas dan kompatibilitas inilah yang banyak menarik minat para programmer untuk menggunakan framework CakePHP sebagai dasar untuk pengembangan aplikasi mereka.

e. Konsep CRUD terintegrasi

CakePHP menerapkan konsep CRUD (Create, Read, Update, Delete) terintegrasi yang membantu interaksinya dengan database dan menyederhanakan query.

f. Arsitektur OOP dan MVC

Kelas-kelas yang menjadi dasar dari CakePHP ditulis dengan konsep Object Oriented Programming yang memudahkan programmer untuk melakukan penambahan, pengurangan dan modifikasi kelas dan fungsi yang digunakan. CakePHP menggunakan arsitektur Model, View, Controller (MVC) yang memisahkan database logic (Model), presentation logic (View) dan bussiness logic (Controller).

g. Fitur Scaffolding

Ya, barangkali inilah fitur yang jarang didapat pada framework lain. CakePHP mempunyai fitur yang mampu men-generate prototipe aplikasi Sebelum menyusun source code-nya secara lengkap. Cukup tambahkan variabel $scaffold di controller, aplikasi web sudah bisa dinikmati.

h. Manajemen akses bagi user

CakePHP memungkinkan pengaturan user dan hak aksesnya dalam aplikasi yang dikembangkan, dengan sarana yang lebih mudah dipahami. Fitur ini dikenal dengan nama Access Control List (ACL).

i. Validasi dan sanitasi data

CakePHP mempunyai kelas-kelas dasar yang membantu programmer untuk melakukan sanitasi dan validasi data pada aplikasinya.

j. Komponen Security, Session, dan Request Handling yang terintegrasi

CakePHP menyediakan komponen-komponen untuk menangani masalah Session, Keamanan (Security) dan Request Handling yang sudah terintegrasi dalam kelas dasar CakePHP. Developer cukup menyertakan komponen tersebut pada controller aplikasi dan memasukkan parameter-parameter untuk mendapatkan hasil yang diinginkan.

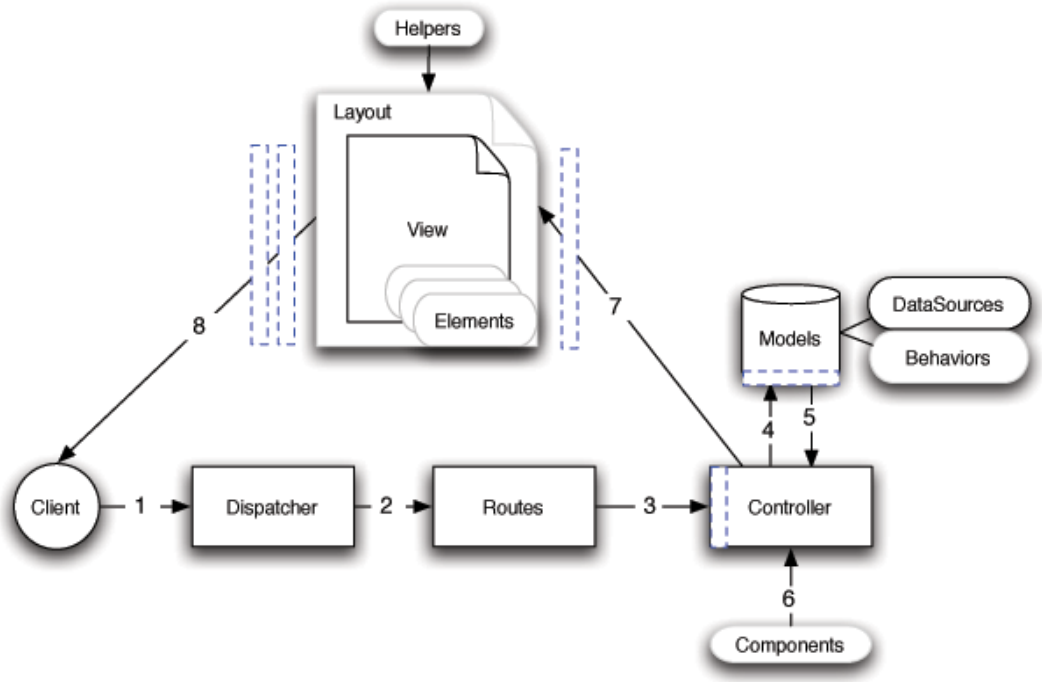
k. Metode templating yang simpel

Sebagaimana framework-framework yang lain, CakePHP juga mendukung metode templating yang sangat mudah digunakan untuk membantu programmer dan web disainer menciptakan tampilan aplikasi yang indah dan mudah dimodifikasi. CakePHP mempunyai kelas helper yang mendukung templating HTML, Ajax, Javascript dan masih banyak lagi.

l. Cocok untuk segala strukutur direktori

CakePHP mempunyai sistem konfigurasi yang menyediakan berbagai macam pilihan konfigurasi, sesuai dengan struktur direktori pengembangan aplikasi berbasis framework CakePHP.

2.9.3 Cara Kerja



Gambar 2.13 Struktur kerja CakePHP

Dari gambar di atas, cara kerja CakePHP sebagian besar sudah bisa dijelaskan.

1. *Client* mengklik *link* ke http://www.example.com/books/beli, dan *browser Client* membuat *request* ke web server.

2. *Route* akan menterjemahkan URL untuk mengekstrak parameter dari *request*, yaitu: *controller*, *action* dan argumen lainnya yang mempengaruhi logik bisnis saat *ruquest*.

3. Dengan menggunakan *route*, URL yang di*request* dipetakan ke *action* di *controller* (sebuah *method* dalam *class controller*). Dalam kasus ini *method* beli() di dalam *class* BooksController. *Callback* beforeFilter() di *controller* dipanggil sebelum logic *action controller* dieksekusi.

4. *Controller* dapat menggunakan *model* untuk memperoleh data aplikasi. Dalam contoh ini, *controller* menggunakan *model* untuk mengambil data pembelian terakhir Joko dari database. Penggunaan *model* tidaklah diperlukan, tapi semua *controller* CakePHP secara inisial membutuhkan setidaknya satu *model*.

5. Setelah *model* memperoleh data, *model* akan memberikannya ke *controller*. *Callback* untuk *model* dapat dipanggil.

6. *Controller* dapat menggunakan *component* untuk memanipulasi data atau melakukan operasi lainnya (misal untuk manipulasi *session*, autentifikasi, atau mengirim email).

7. Saat *controller* selesai menggunakan *model* dan *component* untuk menyiapkan data, *controller* dapat memberikan data tersebut ke *view* dengan pemanggilan *method* set(). *Callback* di *controller* dapat dieksekusi sebelum data dikirim. Kemudian logik di *view* dieksekusi, *element*

dan/atau *helper* dapat disertakan (*include*). Secara default, *view* direnderdi dalam *layout*.

8. Kemudian *callback controller* tambahan (seperti afterFilter()) dapat dipanggil. Setelah itu kode *view* yang direnderdiberikan ke peraba *Client*.

2.9.4 Struktur Aplikasi

```
/app
    /config          - Contains config files for your database, ACL, etc.

    /controllers     - Controllers go here
        /components  - Components go here

    /index.php       - Allows you to deploy cake with /app as the DocumentRoot

    /models          - Models go here

    /plugins         - Plugins go here

    /tmp             - Used for caches and logs

    /vendors         - Contains third-party libaries for this application

    /views           - Views go here
        /elements    - Elements, little bits of views, go here
        /errors      - Your custom error pages go here
        /helpers     - Helpers go here
        /layouts     - Application layout files go here
        /pages       - Static views go here

    /webroot         - The DocumentRoot for the application
        /css
        /files
        /img
        /js

/cake                - Cake's core libraries. Don't edit any files here.

index.php

/vendors             - Used for server-wide third-party libraries.

VERSION.txt          - Let's you know what version of Cake you're using.
```

Gambar 2.14 Struktur Folder CakePHP

Folder /app adalah folder di mana aplikasi web akan dikembangkan. Folder ini berisi beberapa folder lainnya, antara lain :

a. /config : berisi *file-file* konfigurasi CakePHP, termasuk konfigurasi *database, debug option,* ACL *config,* dan *routing option*.
b. /controllers : berisi *file-file controller* yang akan berperan sebagai *bussiness logic* dari aplikasi.
c. /models : berisi *file-file* model yang akan berperan sebagai database *logic* dari aplikasi.
d. /plugins : folder ini digunakan apabila seorang developer ingin mengembangkan aplikasi kecil sebagai pendukung aplikasi utama. *Folder* ini berfungsi sebagai tempat penyimpanan aplikasi pendukung tersebut.
e. /tmp : digunakan untuk fungsi caching dan logging aplikasi
f. /vendors : digunakan apabila programmer melibatkan beberapa aplikasi-aplikasi tambahan (*third-party application*) yang dikembangkan oleh developer lainnya.
g. /views : berisi file - file *view* yang akan berperan sebagai *presentation logic* dari aplikasi Anda. *Folder* ini juga memiliki beberapa *sub folder* yang digunakan untuk menyimpan untuk menyimpan halaman statis (*folder/pages*), halaman *layout* dasar aplikasi (*folder/layouts*), tampilan tambahan yang fleksibel (*folder/elements*) dan sebagainya.
h. /webroot : folder ini berfungsi sebagai DocumentRoot dari aplikasi yang dibuat. Server akan mengakses file index.php yang diletakkan di folder ini.

Folder /cake berisi kelas dan pustaka dasar CakePHP. Pengembang aplikasi tidak disarankan untuk mengubah *file-file* di *folder* ini. Namun untuk beberapa keperluan, para developer CakePHP biasanya melakukan beberapa modifikasi dan perubahan *source code* pada *file* yang ada di *folder* ini.

Folder /vendors digunakan untuk menyimpan aplikasi *third-party* yang digunakan dalam lingkup yang lebih luas. Pemisahan antara *folder* /app dan folder /cake memungkinkan developer untuk memasang banyak *folder* /app dengan memanfaatkan satu *folder* /cake saja. Hal ini juga memudahkan developer untuk melakukan *update* framework CakePHP, cukup download versi terbaru dan timpakan ke *folder* /cake yang lama tanpa mengganggu folder /app.

2.9.5 Penamaan *File* dan *Class*

a. Model

Nama *class* dari *model* ditulis CamelCase dan *singular*. Contoh penamaan *class* model: Person, BigPerson dan ReallyBigPerson.

Nama tabel untuk *model* CakePHP ditulis dengan *underscore* dan *plural*. Nama tabel untuk *model* pada contoh di atas adalah people, big_people, dan really_big_people.

Foreign key dalam hubungan asosiasi hasMany, belongsTo atau hasOne secara *default* dikenali oleh CakePHP dengan nama *model* terkait (dalam bentuk *singular*) diikuti dengan _id. Jadi jika baker

hasMany cakes, tabel cakes akan merujuk baker dalam tabel bakers melalui *foreign key* baker_id.

Tabel join yang digunakan dalam relasi hasAndBelongsToMany (HABTM) antara beberapa *model* sebaiknya diberi nama tabel *model* yang dijoin secara alpabet (misal apples_zebras, bukan zebras_apples).

Semua tabel *model* dalam CakePHP yang saling berinteraksi (kecuali tabel join) membutuhkan *primary key singular* untuk menidentifikasi setiap row secara unik. Jika Anda menginginkan *model* tabel yang tidak memiliki *field primary key*, seperti row dari join table posts_tags, konvensi CakePHP adalah ada satu *field primary* yang ditambahkan ke tabel.

b. Controller

Nama class *controller* berbentuk *plural*, CamelCase, dan diakhiri dengan Controller. Contoh konvensi penamaan *controller* adalah PeopleController dan LatestArticlesController.

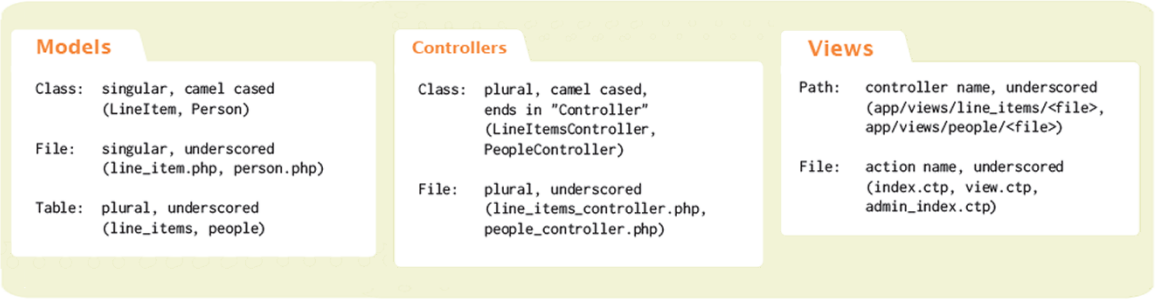
Fungsi pertama yang mungkin ditulis adalah fungsi index(). Saat ada *request* dengan menyertakan *controller* tanpa parameter *action*, CakePHP secara default akan mengeksekusi fungsi index() di *controller* tersebut. Misal, *request* ke http://www.example.com/apples/ akan memanggil fungsi index() di *controller* ApplesController, sedangkan *request* ke http://www.example.com/apples/view/ akan memanggil fungsi view() di *controller* ApplesController.

Anda juga dapat mengubah akses fungsi di *controller* dengan menyisipkan *underscore* di awal nama fungsi tersebut. Jika nama fungsi di *controller* diawali dengan *underscore*, maka fungsi tersebut tidak dapat diakses secara langsung dari web melainkan hanya melalui pemanggilan internal.

c. *View*

File *template View* dinamai berdasarkan fungsi *controllerr yang ditampilkan*, dalam bentuk digaris-bawahi. Fungsi getReady() dari *class* PeopleController akan mencari sebuah *view template* /app/views/people/get_ready.ctp. Pola dasar nya adalah :

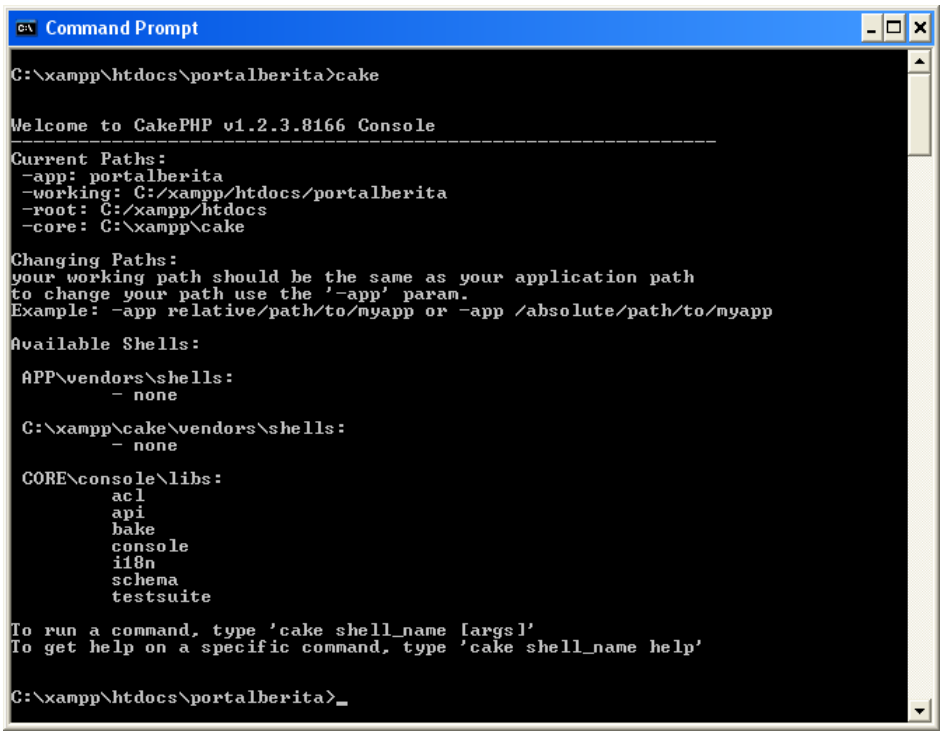
/app/views/controller/underscored_function_name.ctp.



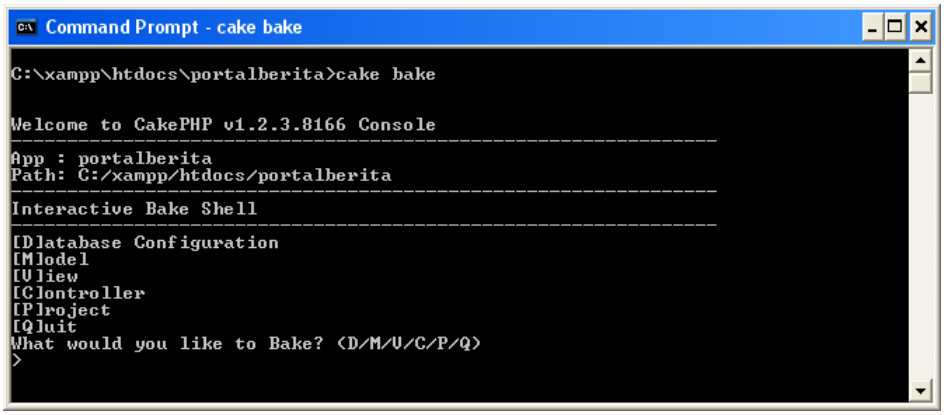
Gambar 2.15 CakePHP Convention

Framework CakePHP menyediakan suatu fasilitas interaktif yang dinamakan *Core Console Application*, berfungsi dalam pembuatan kode-kode CMS secara otomatis sehingga dalam penulisan *source code* jadi lebih mudah.

Program *console* tersebut berada pada folder : `../cake/console` dan dapat dipanggil melalui *shell* seperti dua buah gambar dibawah ini.



Gambar 2.16 CakePHP *Core Console Application*

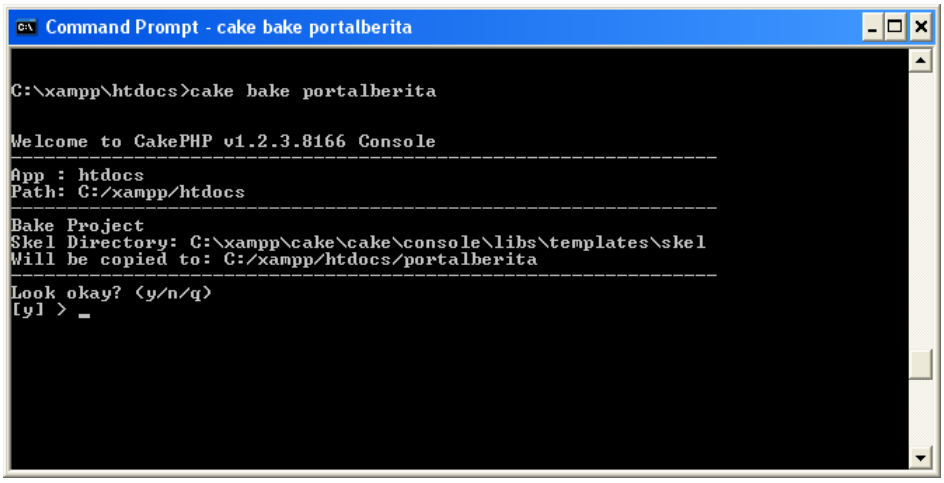


Gambar 2.17 CakePHP *Core Console Application, Bake*

CakePHP menggunakan prinsip *ConventionOverConfiguration*. Maksudnya adalah kita tidak perlu membuat konfigurasi bermacam-macam, namun cukup ikuti saja aturan dari CakePHP agar CakePHP dapat membuat dan membaca configurasi *on the fly* dinamis secara otomatis.

a. Pembuatan *Project*

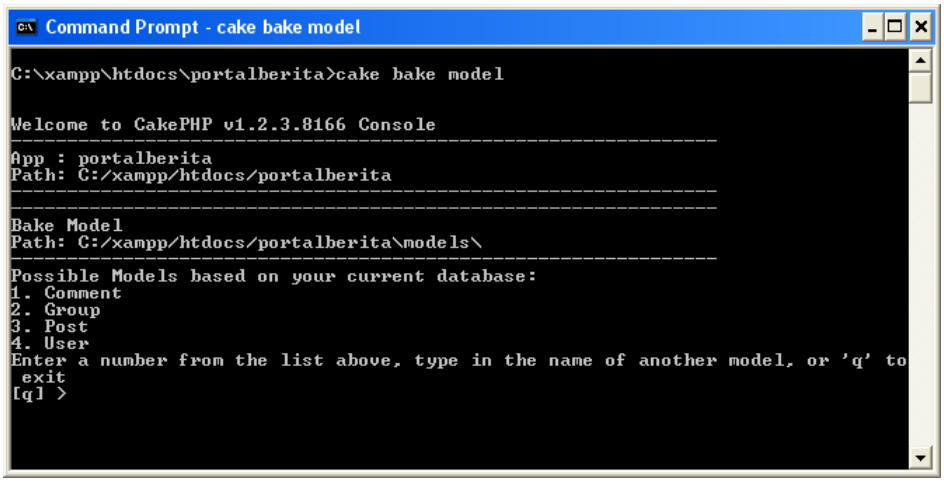
Pada proses ini dilakukan proses konfigurasi awal terhadap lokasi project dan konektifitas nya ke database.



Gambar 2.18 *Baking Project*

b. Pembuatan *Model*

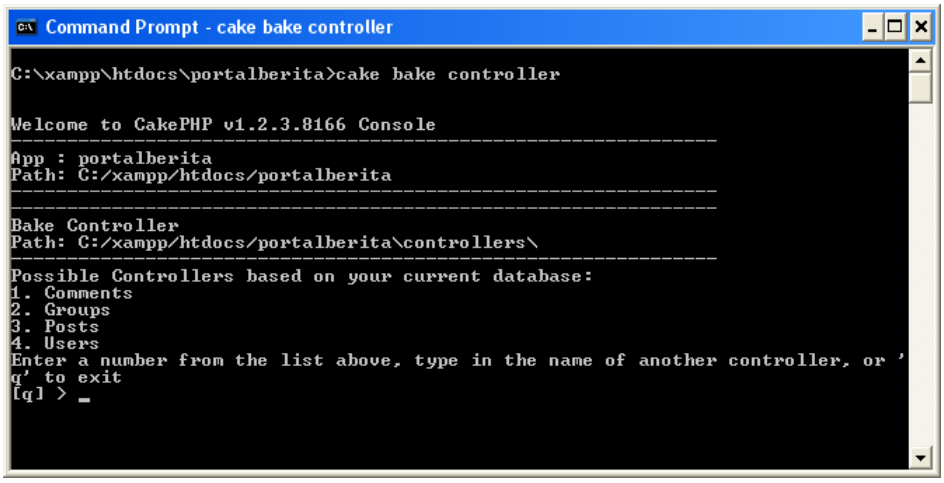
Saat membuat *Model*, cake akan secara otomatis mendeteksi dan memberikan pilihan asosiasi antara *model* (atau relasi antar tabel) yang sesuai.



Gambar 2.19 *Baking Model*

c. Pembuatan *Controller*

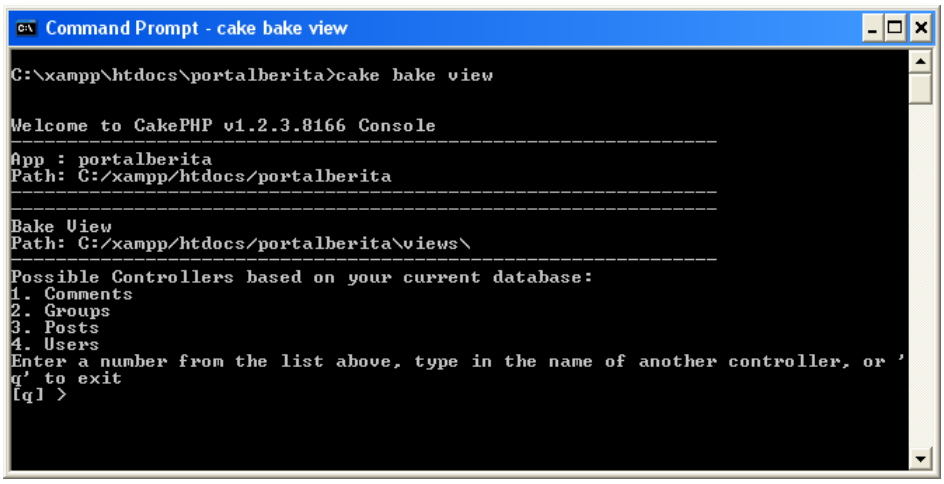
Pembuatan controller pun dapat dilakukan secara interaktif dengan menggunakan core console CakePHP. Model-model yang telah dibuat akan dideteksi terlebih dahulu, dan kemudian dapat dibuatkan controller nya secara otomatis.



Gambar 2.20 *Baking Controller*

d. Pembuatan *View*

Untuk membuat view, terlebih dahulu harus ada model dan controller yang bersangkutan.



Gambar 2.21 *Baking View*

Selain komponen-komponen yang disediakan secara interaktif oleh core console CakePHP, masih banyak lagi fungsi-fungsi lainnya yang telah di-bundle secara built-in kedalam framework CakePHP ini sebagai library. Dimana dokumentasi nya dapat diakses secara online melalui internet pada situs resminya.

2.10 *Hyper Text Markup Language* (HTML)

HTML merupakan kepanjangan dari *Hyper Text Markup Language* adalah suatu bahasa yang digunakan untuk membuat halaman-halaman hypertext (hypertext page) pada internet. Dengan konsep hypertext ini, untuk membaca suatu dokumen tidak harus melakukannya secara urut, baris demi baris, atau halaman demi halaman. Tetapi tidak dapat dengan mudah melompat dari satu

topik ke topic lainnya yang disukai, seperti halnya jika melakukan pada online Help dari suatu aplikasi Windows.

HTML dirancang untuk digunakan tanpa tergantung pada suatu platform tertentu (platform independent).

2.11 PHP

Menurut Luke Welling dan Laura Thomson(2001,p1), PHP adalah *server-side scripting language* yang didesain secara spesifik untuk web. Dalam *page HTML*, dapat dimasukkan *code* PHP yang akan dieksekusi setiap kali halaman dikunjungi. PHP *code* diterjemahkan di *web-server* dan dirubah menjadi HTML atau output lain yang akan dilihat oleh pengunjung halaman.

PHP diciptakan oleh Rasmus Lerdorf, seorang pemrogram C yang handal. Semula PHP hanya digunakan untuk mencatat jumlah pengunjung pada homepagenya. Rasmus adalah seorang pendukung open source. Karena itulah ia mengeluarkan Personal Home Page Tools versi 1.0 secara gratis. Setelah mempelajari YACC dan GNU Bison, Rasmus menambah kemampuan PHP 1.0 dan menerbitkan PHP 2.0.

PHP mudah dibuat dan cepat dijalankan, PHP dapat berjalan dalam web server yang berbeda dan dalam sistem operasi yang berbeda pula. PHP dapat berjalan di sistem operasi UNIX, Windows 98, Windows XP, Windows NT, dan Macintosh.

Kelebihan PHP :

a. Bahasa pemrograman PHP adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaanya.
b. Web Server yang mendukung PHP dapat ditemukan dimana - mana dari mulai IIS sampai dengan apache, dengan konfigurasi yang relatif mudah.
c. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.
d. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena referensi yang banyak.

PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (linux, unix, windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah system.

2.12 Ajax

Asynchronous JavaScript and XMLHTTP, atau disingkat AJaX, adalah suatu teknik pemrograman berbasis web untuk menciptakan aplikasi web interaktif. Tujuannya adalah untuk memindahkan sebagian besar interaksi pada komputer web surfer, melakukan pertukaran data dengan server di belakang layar, sehingga halaman web tidak harus dibaca ulang secara keseluruhan setiap kali seorang pengguna melakukan perubahan. Hal ini akan meningkatkan interaktivitas, kecepatan, dan usability. Ajax merupakan kombinasi dari:

a. DOM yang diakses dengan client side scripting language, seperti VBScript dan implementasi ECMAScript seperti JavaScript dan JScript, untuk menampilkan secara dinamis dan berinteraksi dengan informasi yang ditampilkan.
b. Objek XMLHTTP dari Microsoft atau XMLHttpRequest yang lebih umum di implementasikan pada beberapa browser. Objek ini berguna sebagai kendaraan pertukaran data asinkronus dengan web server. Pada beberapa framework AJAX, element HTML IFrame lebih dipilih daripada XMLHTTP atau XMLHttpRequest untuk melakukan pertukaran data dengan web server.
c. XML umumnya digunakan sebagai dokumen transfer, walaupun format lain juga memungkinkan, seperti HTML, plain text. XML dianjurkan dalam pemakaian teknik AJaX karena kemudahan akses penanganannya dengan memakai DOM.
d. JSON dapat menjadi pilihan alternatif sebagai dokumen transfer, mengingat JSON adalah JavaScript itu sendiri sehingga penanganannya lebih mudah.

Seperti halnya DHTML, LAMP, atau SPA, Ajax bukanlah teknologi spesifik, melainkan merupakan gabungan dari teknologi yang dipakai bersamaan. Bahkan, teknologi turunan/komposit yang berdasarkan Ajax, seperti AFLAX sudah mulai bermunculan.

2.13 MySQL

MySQL adalah salah satu jenis database server yang sangat terkenal. Kepopulerannya disebabkab bersifat free (tidak perlu membayar untuk menggunakannya) pada berbagai platform (kecuali pada windows, yang bersifat software atau perlu membayar setelah melakukan evaluasi dan memutuskan digunakan untuk keperluan produksi). Perangkat lunak MySQL sendiri bisa di download dari http: // www.mysql.org atau http://www.mysql.com.

MySQL termasuk jenis RDMS *(Relational Database Management System)*. Itulah sebabnya istilah seperti tabel, baris, dan kolom digunakan pada MySQL. Pada MySQL, sebuah database mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom.

Menurut Menurut Luke Welling dan Laura Thomson(2001,p1), MySQL adalah sebuah relational database management systems yang sangat cepat dan kuat.

MySQL adalah sebuah perangkat lunak sistem management basis data SQL atau DBMS yang multithread, multiuser, dengan sekitar 6 juta instalasi diseluruh dunia. MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius. Tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General

Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Untuk melakukan administrasi dalam basis data MySQL, dapat menggunakan modul yang sudah termasuk yaitu *command-line* (perintah: mysql dan mysqladmin). Juga dapat diunduh dari situs MySQL yaitu sebuah modul berbasis grafik (*GUI*): *MySQL Administrator* dan *MySQL Query Browser*. Selain itu terdapat juga sebuah perangkat lunak gratis untuk administrasi basis data MySQL berbasis web yang sangat populer yaitu phpMyAdmin.