**BAB III**

**PERANCANGAN DAN PEMBUATAN FRAMEWORK**

**3.1. Bahan dan Alat**

Dalam penelitian ini dibutuhkan bahan dan alat penelitian. Software yang dibutuhkan dalam pembuatan aplikasi ini antara lain :

1. *Web Server*

*Web server* yang digunakan untuk pembuatan *framework* ini adalah Xampp 1.7.3 yang sudah terdapat didalamnya antara lain *apache, php dan mysql.*

1. *Browser*

*Browser* yang penulis gunakan untuk menguji aplikasi adalah *google chrome* dan *mozilla* *firefox*.

Sedangkan *hardware* yang digunakan dalam pembuatan *framework* adalah :

1. Monitor 14 inci
2. Processor Intel Core 2 Duo 2.0 GHz
3. Harddisk 160GB
4. DDRAM 3GB

*Hardware* minimum yang dapat digunakan dalam pembuatan *framework* adalah :

1. Monitor 14”
2. Processord Pentium 4
3. Harddisk minimal 40GB
4. DDRAM 512MB

**3.2. Kerangka Kerja (*Framework) PHP***

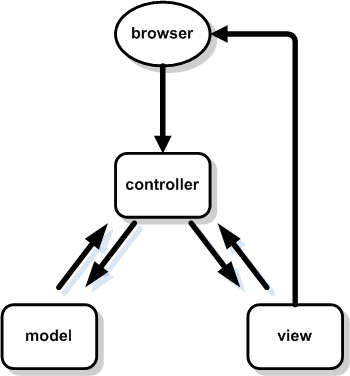
Membuat sebuah website adalah menghubungkan banyak file .html , .php , .asp sehingga menjadi satu kesatuan tentunya dengan mengikuti kaidah-kaidah pembangunan *website* yang berlaku. Pembuatan *website* yang seperti itu masih menggunakan cara konvensional dengan membuat masing - masing *template* untuk setiap *file*. Jadi *design* dan *programming* gabung dalam satu *file*, kemudian diulangi lagi untuk *file* yang berikutnya dan begitu terus untuk semua *file*.

Ada beberapa konsep dalam membuat *framework,* salah satunya adalah MVC (*Model View Controller*).

MVC itu sendiri adalah konsep pemrograman yang memisahkan pemrograman *logic* aplikasi dengan presentasinya. Perkembangan *mvc* dalam dunia pemrograman adalah  perkembangan dimana setiap orang bisa belajar dan membuat program yang *enterprise* (yang berbobot) tanpa harus mahir dan belajar bertahun - tahun atau pun membutuhkan *teamwork* yang banyak. konsep *mvc* tidak hanya di gunakan dalam satu jenis program, bahkan sekarang bisa dikatakan hampir menyeluruh di setiap jenis pemrograman, baik yang open source maupun yang berbayar.

**3.2.1. Rancangan Konsep Model View Control**

Coding yang didasari arsitektur MVC, maka proses *development software* memiliki pondasi yang kuat untuk mengarah ke aplikasi dengan skala besar. Karena semua *class library* sudah diatur sesuai dengan fungsinya masing-masing yang mudah dipelajari dan digunakan oleh siapapun programmernya. Oleh karena itu banyak sekali yang menggunakan metode ini untuk mengembangkan suatu proyek *website* yang sangat kompleks.

****

**Gambar 3.1. Diagram MVC**

**3.3. Pembuatan Kerangka Kerja (*Framework*)**

Tata cara pembuatan *Controller*, *View* dan struktur lainnya seperti *Loader*, Register Object dan sebagainya.

**3.3.1. index.php**

*File* ini merupakan *file* utama yang nantinya selalu diakses oleh siapapun yang memanggilnya. Walaupun *file* utama tetapi isi programnya sangat sederhana karena pada dasarnya *file* ini hanya berisi konstanta - konstanta yang dibutuhkan oleh sistem.

**3.3.2. system**

Program utama dari *framework* ini. Pada umumnya sebuah *framework* selalu dilengkapi dengan sistem inti (*core*), *library* dan *helper*. Sistem *core* itu sendiri sebenarnya hanya ada satu *class* utama yaitu *controller*. Tugas utama *controller* adalah memuat (*loader*) *object library*, *helper*, *database*, *view* dan *object* pendukung lainnya.

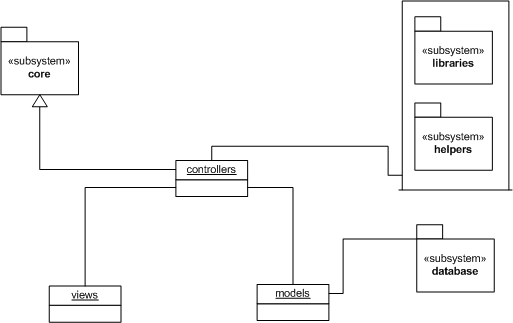
*controller* sendiri dibagi menjadi 3 *class* :

* *class loader*
* *class registry*
* *class controller*

**3.3.3. Application**

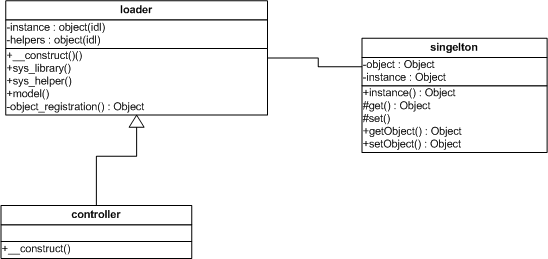
Folder *application* adalah lokasi program. Dalam *directory* tersebut terdapat *sub directory controllers*, *models* dan *views*. *Sub directory controllers* tersebut nantinya bisa diisi dengan *class - class controller*. *Directory models* berisi *class - class* pengolahan database seperti *insert*, *update*, *delete* dan *select*. Sedangkan *views* digunakan untuk menyimpan *tag - tag HTML*.

**3.4. Struktur Kerja Framework**

 Secara garis besar cara kerja framework kita ini dapat dilihat pada gambar dibawah ini.

**Gambar 3.2. Struktur Kerja *Framework*.**

**3.4.1. *Sub System Core***

*Sub sistem core* adalah kerangka utama dari keseluruhan *framework* yang akan dibangun. Pada *sub sistem core* ini terdapat 3 class utama, yaitu : *class loader, class singleton* dan *class controller*.

**Gambar 3.3. Susunan *class* diagram dari ketiga *class***

**3.4.1.1. *Class Controller***

*Class controller* adalah *class* yang paling pokok dalam sistem core tersebut. Class controller inilah nantinya akan diturunkan (*extends*) ke semua *class controller* yang dibuat oleh pengembang *web site*. Yang Perlu dipahami bahwa tugas utama *class controller* adalah memuat (*loaded*) semua *object* yang dibutuhkan oleh sistem, seperti misalnya *library, helper, database, view, plugins* dan lain sebagainya. Agar *coding* tidak terlalu panjang maka fungsi - fungsi untuk memuat berbagai macam *object* tersebut dituliskan kedalam *class loader*.

**3.4.1.2. *Class Loader***

Tugas *class loader* adalah memuat berbagai macam *object* yang dibutuhkan oleh sistem. Namun kenyataannya pada saat memuat sebuah *object*, bisa saja *object* yang sama sudah pernah termuat sebelumnya. Jika hal ini dibiarkan maka akan mengakibatkan terjadinya penumpukan (*stack*) inisialisasi *object* yang sama pada memory. Penumpukan inisialisasi *object* ini tentu saja mengakitbatkan pemborosan pemakaian *memory*. Untuk menghindari hal tersebut maka dibuatlah *class singleton*.

**3.4.1.3. *Class Singleton***

Tugas utama *class singleton* adalah memastikan bahwa tidak ada *object* yang sama dimuat lebih dari satu kali. Penghematan penggunaan *memory* wajib kita lakukan dengan harapan manakala *web site* kita sudah diakses oleh banyak orang, maka bisa kita minimalisir kemungkinan terjadinya macet (*down*) pada *web server*.

**3.4.2. *Controller***

*Controller* adalah *file* yang menghubungkan *requset* dari *view* ke *model*. Di dalam *controller* ini pula dilakukan pengolahan *logic* sebelum dikirimkan ke *file view* untuk ditampilkan ke *user*.

**3.4.2.1. Mengakses *Controller***

*Controller* sebuah *class* yang dalam mengaksesnya menggunakan *path url* tertentu. Nama *path url* mewakili nama *controller*.

Contoh :

<http://localhost/cp27/index.php/welcome>

Maka yang menjadi *controller*-nya adalah ***welcome.***

**3.4.2.2. *Default Controller***

Jika pada alamat *url* tidak ditambahkan argumen nama *controller*, otomatis akan mengakses *controller default* yang bernama ***home***.

Contoh :

<http://localhost/cp27/>

Hasilnya akan sama dengan:

<http://localhost/cp27/index.php/home>

**3.4.3. Model**

Model adalah sebuah *class* yang tugasnya berkaitan langsung dengan penanganan data, baik itu dari *database*, atau sistem penyimpanan yang lain.

**3.4.3.1. Membuat Model**

Untuk membuat sebuah *class* model, tambahkan file baru dan letakan di dalam *folder* **apps/model/**, misalnya **users.php**.

Buat *class users* di dalam file ini dengan diawali *prefix* **"Model\_"**, contohnya:

<?php

class Model\_users {

public function \_\_construct(){

}

}

**3.4.3.2. Load Library di dalam Model**

me-*load library* atau *class* lain di dalam model ini, ada dua cara yang bisa dilakukan. Pertama, jika *resource* dari *class* atau *library* tersebut ingin digunakan oleh semua *method* yang ada di dalam *class* model, maka pengambilan dan penginisialisasian bisa dilakukan di *method* **\_\_construct**. Namun, jika *resource* hanya ingin digunakan pada salah satu *method* saja, maka inisialisasi dan pemanggilan bisa dilakukan di dalam *method* yang bersangkutan. Salah satu hal yang paling umum dilakukan di dalam sebuah *class* model adalah me-*load library database*. Berikut adalah contohnya:

<?php

class Model\_users {

public function \_\_construct(){

$this->db = new Library\_db();

}

}

**3.4.4. *View***

*View* adalah sebuah *file* yang bertugas untuk menampilkan ***output*** dari semua proses yang sudah dilakukan pada bagian [*controller*](http://localhost/cms/panada/documentation/id/references/controller.html) dan [*model*](http://localhost/cms/panada/documentation/id/references/model.html).

**3.4.4.1. Membuat View**

*Folder* *view* terletak pada:

apps/view/

Buat file *php* dengan nama *helloworld.php* di dalam *folder* ini.

<html>

<head></head>

<body>Hello world!</body>

</html>

Untuk bisa digunakan di dalam aplikasi, *file view* harus di-*load* di dalam *method controller*. Berikut adalah contoh me-*load file view* di dalam *method index* pada *controller home*.

<?php

class Controller\_home extends Panada {

public function \_\_construct(){

parent::\_\_construct();

}

public function index(){

$this->view\_helloworld();

}

}

**3.4.5. *Library***

*Library* adalah sebuah *class* yang memiliki tugas spesifik yang fungsinya bisa digunakan oleh *controller*, *model* ataupun *library* lainnya.

**3.4.5.1. Membuat *Library***

*Library* bisa di simpan di dalam *folder*

apps/library/

Yang harus diperhatikan dalam membuat *library* adalah, setiap nama *class* harus diawali dengan *prefix* **"Library\_"**.

<?php

class Library\_register {

public function \_\_construct(){

}

}