

CakePHP översikt samt tutorial

Vad är CakePHP	2
Ett exempel på namngivnings konvention i Cake.....	2
Cakes byggstenar	3
View	3
Model	3
Controller	3
Routing:.....	4
Tutorial	5
1. Ladda hem CakePHP.....	5
2. Installation	5
3.Surfa till din blog genom http://127.0.0.1:8088/blog/	5
4 Databas anslutning.....	6
5 Skapa din Post Model	7
6 Skapa Controller.....	7
7 Skapa index vy.....	8
8 Routing	8
9 Index View cd.....	8
10 View Controller och vy	9
11 Add Controllen, add vy och validering	10
12 Delete vy och controller	11
13Edit vy och controllern	12

Vad är CakePHP

CakePHP är ett open source PHP ramverk och skapades år 2005. Cake har som ambition att hjälpa till att skapa komplexa webbplatser 'rapidly'. Cake använder sig av MVC design mönster och är objekt orienterad.

För att få ut det optimala ur ramverket måste man använda sig av vissa konventioner vid namngivningen av sina klasser, mappar och funktioner. Gör man det erbjuder Cake en bredd gamma av funktionalitet bland annat: code generering (m.h.a. cake console), integrerad CRUD funktionalitet, validering, Access Control SEO vänliga URL och mycket annat.

Cakes koncept är ganska enkelt men mycket kraftfullt; gör man som det står i Cakes guider och online dokumentationen är det svårt att misslyckas.

I skrivandes stund är det version 2.0.5 som är det nyaste och publicerades den 29/12/2011. Det är väldigt populärt att använda sig av version 1.3 av Cake också. Ramverket är mycket populärt och det finns en stor aktiv grupp av användare och utvecklare som är mycket hjälpsamma.

Denna version har inbyggd stöd för följande databaser:

Database/Mysql - MySQL 4 & 5,
Database/Sqlite - SQLite (PHP5 bara),
Database/Postgres - PostgreSQL 7 och högre,
Database/SqlServer - Microsoft SQL Server 2005 och högre

Ett exempel på namngivnings konvention i Cake.

Man ska namnge sin databastabell i plural; tex en tabell i vilken vi ska ha blog poster kommer att heta posts.

Model

filnamn = Order.php

class = Order extends AppModel

directory = Model

Controller

filnamn = PostsController.php

classname = PostsController extends AppController

directory = Controller

View

filnamn = samma som controllens action

extension = .ctp

directory = View/Posts

Standard actions i kontrollen:

index();

add();

edit();

view();

delete();

Mot detta actions använder vi standard vyer i View/Posts map:

index.ctp
add.ctp
edit.ctp
view.ctp
delete.ctp

Cakes byggstenar

View

View är till för att göra model data synlig för slutanvändaren. Data kommer inte direkt från Model till View, det transporteras genom Controllen. Controllen väljer rätt View beroende på vad en användare vill göra med modellen. Controllen vet det eftersom användaren anropar en action i Controllen. View är som grafisk interface på data modellen. Själva vyn är kan inte göra något själv så den behöver Model för att ha något vettigt att säga och Controllen för att veta när den ska säga det.

Alla vyer befinner sig i undermappen View/Modelens-namn-i-plural/action-i-controllen-namn och har extension .ctp.

Om vi tex har model som heter Post kommer dess add view (lägga till en ny post) se ut på följande sätt:

../View/Posts/add.ctp och URL som användaren kommer att behöva klicka på för att komma åt den kommer att se ut så här:

<http://exemple.com/blogprojekt/posts/add>

Detta kan individuellt anpassas men detta är det rekommenderade sättet.

Model

Model är det den delen av applikationen som 'vet saker'. Det är data som applikationen livnär sig på. Representeras ofta av en databas där varje tabell representerar ett objekt class och varje dess row dess instance. Man kan manipulera data, lägga till nya objekt, förändra dessa eller ta bort totalt. Objekt har ofta relationer med andra objekt.

Controller

Här har vi en riktig smart kille som styr och säger alla andra vad och när dem ska göra saker för att inte göra bort sig. Controllen hämtar data från Modellen och sedan säger han till View att visa just detta data. Controller använder sig av actions för att anropa rätt vy för att presentera model.

Så här kan index funktion se ut:

```
public function index() {  
  
    $this->set('posts', $this->Post->find('all'));  
}
```

Denna raden hämtar alla Post objekt från databasen och skickar dessa till view med hjälp av set. All information om vårt objekt kommer nu vara tillgänglig i filen View/Posts/index.ctp och vi kan

använda dess kunskap genom att använda sig av variabeln \$posts i vår vy för att skriva ut data vi behöver.

Routing:

Routing är viktig i Cake, den hjälper oss att välja hur vi får tillträde till våra filer. I detta exempel kopplar vi ihop base path ('/') till Posts controlen och dess index action;

```
Router::connect('/', array('controller' => 'posts', 'action' => 'index'));
```

Detta gör vi i ../Config/routes.php filen.

Cake vet då att det är index.ctp som ska anropas för att skriva ut modelens data.

Tutorial

Det är enkelt att komma igång med Cake och här kommer en tutorial som beskriver hur man kan skapa en enkel blog med hjälp av Cake.

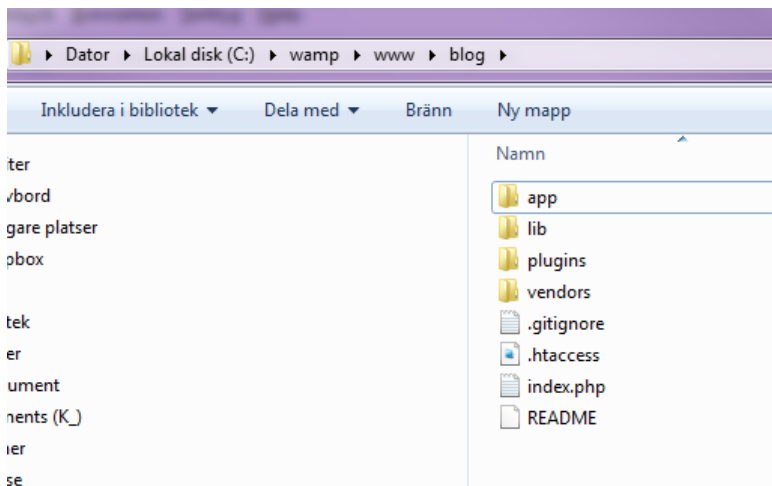
1. Ladda hem CakePHP

CakePHP finns på <https://github.com/cakephp/cakephp/zipball/2.0.5> och hämtar version 2.0.5 av Cake som används i denna tutorial.

2. Installation

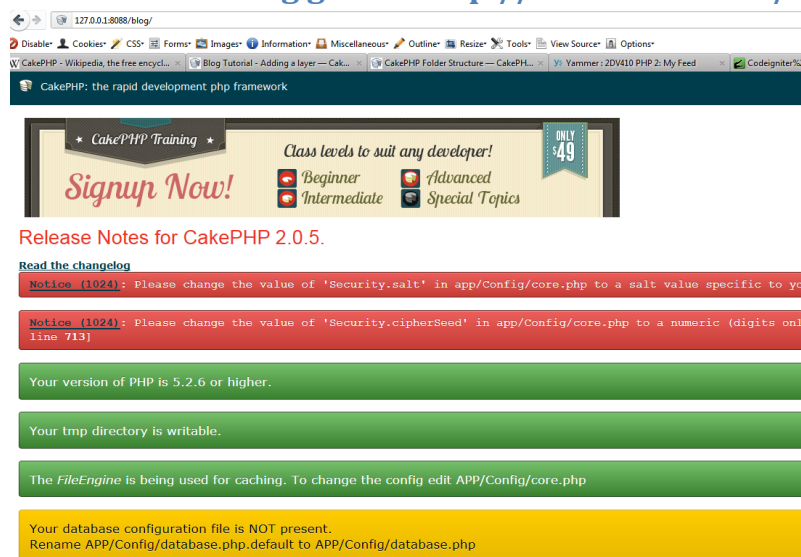
Installera Cake genom att packa upp den nyss nerladdade zip filen i rooten av ditt projekt katalog. Jag har alla mina projekt i C:\wamp\www och det är i www katalogen jag har valt att packa upp Cake. Efter det byter jag namn på den uppackade filen till det namnet jag vill; här använder jag mig av blog.

Detta der ut på följande sätt:



Navigera till app folder; det är i den allt du behöver befinner sig.

3.Surfa till din blog genom <http://127.0.0.1:8088/blog/>



[Editing this Page](#)

Det betyder att du har installerat Cake lokalt men du måste ändra Security.salt, Security.cipherSeed samt konfigurera en anslutning till databasen.

Security.salt, Security.cipherSeed värden finns i app/Config/core.php. Ändra dessa; lättast är det att gå till: http://www.sethcardoza.com/tools/random_password_generator och generera nya värden för att senare ändra i våra filer.

4 Databas anslutning

Du måste förbereda en databas för att vårt projekt ska kunna använda den. Skapa en databas 'cakeblog' table m.h.a. till exempel phpmyadmin. Där skapa tabellen posts som ska vara grunden för vår model och lägg till några exempel poster.

```
CREATE TABLE posts (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(50),
    body TEXT,
    created DATETIME DEFAULT NULL,
    modified DATETIME DEFAULT NULL
);

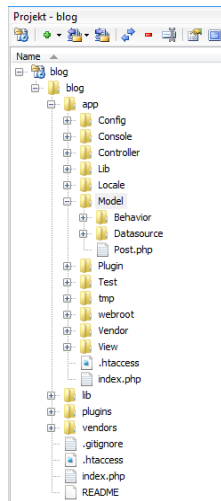
/* exempel värden */
INSERT INTO posts (title,body,created)
VALUES ('Min första post, Detta är body på min post.', NOW());
INSERT INTO posts (title,body,created)
VALUES ('Ett nytt inlägg', Min andra post.', NOW());
```

Öppna /app/Config/database.php.default och spara den som /app/Config/database.php; fyll i dina databas uppgifter. Hos mig ser det ut på följande sätt:

```
public $default = array(
    'datasource' => 'Database/Mysql',
    'persistent' => false,
    'host' => 'localhost',
    'login' => 'root',
    'password' => "",
    'database' => 'cakeblog',
    'prefix' => "",
    'encoding' => 'utf8',
);
```

5 Skapa din Post Model

I app/Model mapen skapa en Post.php fil.



Eftersom Cake tar hand om det mesta så är Post.php väldigt enkel. Just nu räcker det att den ser ut på följande sätt:

```
<?php
class Post extends AppModel {
    public $name = 'Post';
}
OBS! public $name = 'Post'; används för PHP 4 kompalibitet.
```

6 Skapa Controller

Skapa PostsController.php fil i app/Controller mapen.

Grundstommen ser ut på följande sätt:

```
<?php
class PostsController extends AppController {
    public $name = 'Posts';
}
```

Det är i denna klass som vi ska placera våra action som tillåter våra användare att påverka vår applikation.

Vi börjar med index() som är en default action som anropas när vår användare anropar vår controller.

I index metoden använder vi följande kod rad för att hämta alla poster och skicka info om dessa till vår vy.

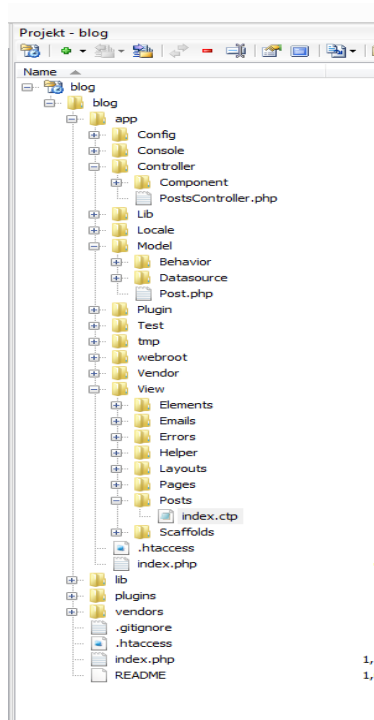
```
$this->set('posts', $this->Post->find('all'));
```

Nästa steg är att skapa en vy som vi kan förse med data och visa för användarna.

7 Skapa index vy

För att kunna visa det hämtade data för användaren måste vi i app/View folder skapa en ny folder som heter som vår model i plular dvs app/View/Posts och i den skapa en tmplate fil index.ctp som kommer visas varje gång controllens index action anropas.

Hela fil strukturen borde se ut på följande sätt:



Skulle vi skriva

```
<?php
```

echo Test; i vår index.ctp och surfa till <http://127.0.0.1:8088/blog/posts> så borde vi se applikationen eka ut Test.

Eftersom vi vill att våra användare inte ska behöva skriva <http://127.0.0.1:8088/blog/posts> och bara <http://127.0.0.1:8088/blog> ska vi nu göra lite ändringar i vår routing.

8 Routing

Gå till app/Config/routes.php och ändra rad 28

```
Router::connect('/', array('controller' => 'pages', 'action' => 'display', 'home'));
```

till följande:

```
Router::connect('/', array('controller' => posts, 'action' => 'index'));
```

Den lilla ändringen kommer att direkt skicka våra användare till PostsControllen och des index action.

9 Index View cd

Vi vill inte att bara eka ut hårdkodade strängar; som vi kommer ihåg skickade vi med hjälp av vår controller information om vår data. Detta är rätt tidpunkt att utnyttja detta.

I vår index.ctp kan vi loopa genom våra poster och visa dessa för våra användare.

```
<h1>Blog posts</h1>
<table>
    <tr>
        <th>Id</th>
        <th>Title</th>
        <th>Created</th>
    </tr>

    <!-- Here is where we loop through our $posts array, printing out post
    info -->

    <?php foreach ($posts as $post): ?>
    <tr>
        <td><?php echo $post['Post']['id']; ?></td>
        <td>
            <?php echo $this->Html->link($post['Post']['title'],
array('controller' => 'posts', 'action' => 'view', $post['Post']['id']));
?>
        </td>
        <td><?php echo $post['Post']['created']; ?></td>
    </tr>
    <?php endforeach; ?>

</table>
```

Surfar vi till vår starta sida kommer vi nu att kunna se våra poster.

Denna kodraden:

```
<?php
echo $this->Html->link($post['Post']['title'],
array('controller' => 'posts', 'action' => 'view', $post['Post']['id']));
?>
```

gör det möjligt att klicka på postens titel för att komma åt dess detaljerad vy. Försöker vi klicka på länken just nu kommer vi att få ett felmeddelande. Det finns ingen view action i vår Controller och ingen view vy att visa heller.

10 View Controller och vy

För att kunna se den enskilda posten måste vi skapa en view action i vår PostsController.php fil.

Denna action/metod kan se ut på följande sätt:

```
public function view($id = null) {
    $this->Post->id = $id;
    $this->set('post', $this->Post->read());
}
```

Men som vi kommer ihåg ger oss det inte så mycket om vi inte kan visa vårt data för användaren. För att göra detta måste vi skapa en view.ctp fil i app/View/Posts/ mapen.

Denna vy kan se ut på följande sätt:

```
<h1><?php echo $post['Post']['title']; ?></h1>

<p><small>Created: <?php echo $post['Post']['created']; ?></small></p>

<p><?php echo $post['Post']['body']; ?></p>
```

Klickar vi på vår länk nu kommer vi kunna se detaljerad information om vår post.

Kollar vi vår URL nu så kommer den se ut på liknande sätt:

<http://127.0.0.1:8088/blog/posts/view/6>; nr 6 är det id nummer som vår post har i databasen.

11 Add Controllen, add vy och validering

För att kunna lägga till nya poster måste vi börja med att lägga till en ny komponent i vår controller.

Detta gör vi genom att lägga till följande rad i Controller.

```
public $components = array('Session');
```

Sedan måste vi skapa en action som tillåter skapande av nya poster. Vi behöver en funktion som ser ut på följande sätt:

```
public function add() {
    if ($this->request->is('post')) {
        if ($this->Post->save($this->request->data)) {
            $this->Session->setFlash('Your post has been saved.');
```

```
            $this->redirect(array('action' => 'index'));
        } else {
            $this->Session->setFlash('Unable to add your post.');
```

```
        }
    }
}
```

vi behöver också en vy som heter add.ctp

```
<h1>Add Post</h1>
<?php
echo $this->Form->create('Post');
echo $this->Form->input('title');
echo $this->Form->input('body', array('rows' => '3'));
echo $this->Form->end('Save Post');
?>
```

För att vara säker att inga tomma rader skickas till databasen måste vi validera input innan vi sparar det. Detta gör vi i våran model klass dvs i app/Model/Post.php filen.

I dess klass lägger vi till följande array med validerings regler:

```
public $validate = array(
    'title' => array(
        'rule' => 'notEmpty'
    ),
    'body' => array(
        'rule' => 'notEmpty'
    )
);
```

Detta gör att man kan inte skicka in tomma fält. Surfar vi nu till:

<http://127.0.0.1:8088/blog/posts/add> kommer vi se följande vy:

CakePHP: the rapid development php framework

Add Post

Title*

Body*

Save Post

Denna raden i vår controller `$this->redirect(array('action' => 'index'))`; gör att vi strax efter att vi har lyckats att lägga till en ny post kommer vi omdirigerade till controllens index action.

12 Delete vy och controller

Det kan hända ibland att man vill ta bort sina poster utan att behöva gå direkt till databasen. Detta kan vi göra genom att implementera delete funktionalitet i vår applikation.

I vår controller måste vi skapa följande action:

```
<?php
function delete($id) {
    if ($this->request->is('get')) {
        throw new MethodNotAllowedException();
    }
    if ($this->Post->delete($id)) {
        $this->Session->setFlash('The post with id: ' . $id . ' has been
deleted.');
```

`$this->redirect(array('action' => 'index'))`; Ingen vy behövs; bara en länk till själva action metoden som innehåller info om postens id. Detta kan vi åstadkomma genom att vi i index.ctp filen inom foreach loopen lägger till följande kod:

```
<?php
echo $this->Form->postLink('Delete',
    array('action' => 'delete', $post['Post']['id']),
    array('confirm' => 'Are you sure?'));
?>
```

Detta kommer ge oss följande vy:

CakePHP: the rapid development php framework

The post with id: 6 has been deleted.

Blog posts

Id	Title	Action
9	vilken user	Delete
10	mania	Delete
11	kaja	Delete
12	misiso	Delete
13	ble ble	Delete

13 Edit vy och controllern

Vill vi kunna redigera våra poster är måste vi genom att följa logiken i föregående steg lägga till en action som heter edit i vår controller samt en motsvarande vy för detta.

I vår controller lägger vi till edit action:

```
function edit($id = null) {
    $this->Post->id = $id;
    if ($this->request->is('get')) {
        $this->request->data = $this->Post->read();
    } else {
        if ($this->Post->save($this->request->data)) {
            $this->Session->setFlash('Your post has been updated.');
```

```
            $this->redirect(array('action' => 'index'));
        } else {
            $this->Session->setFlash('Unable to update your post.');
```

```
        }
    }
}
```

samt skapar en vy edit.ctp med följande innehåll:

```
<h1>Edit Post</h1>
<?php
    echo $this->Form->create('Post', array('action' => 'edit'));
    echo $this->Form->input('title');
    echo $this->Form->input('body', array('rows' => '3'));
    echo $this->Form->input('id', array('type' => 'hidden'));
    echo $this->Form->end('Save Post');
```

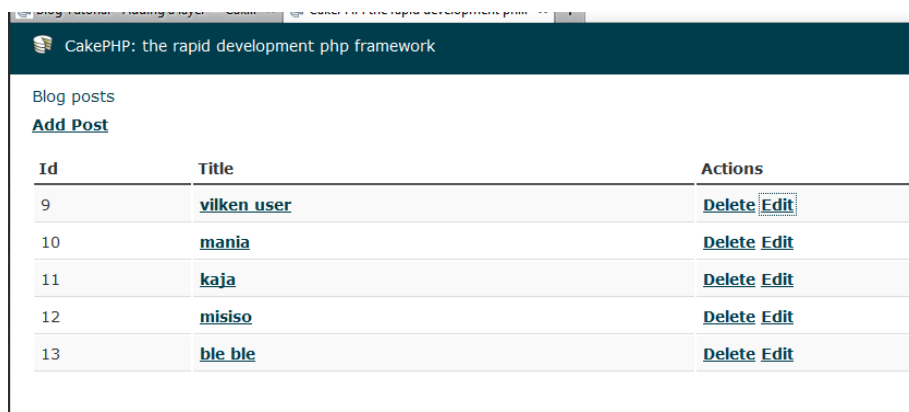
För att vi ska kunna editera våra poster måste vi gå till tex <http://127.0.0.1:8088/blog/posts/edit/9> för att editera post med id 9. Detta kan vara svårt; därför lägger vi till en edit länk i vår index view som kommer skicka oss till rätt post som ska redigeras. Detta gör vi genom att lägga till den kodraden inom vår foreach loop

```
<td>
<?php echo $this->Html->link('Edit', array('action' => 'edit',
$post['Post']['id']));?>
</td>
```

Har vi gjort allt rätt kommer index filen se ut på följande sätt:

```
<h1>Blog posts</h1>
<p><?php echo $this->Html->link('Add Post', array('action' => 'add'));
?></p>
<table>
    <tr>
        <th>Id</th>
        <th>Title</th>
        <th>Actions</th>
        <th>Created</th>
    </tr>
    <?php foreach ($posts as $post): ?>
    <tr>
        <td><?php echo $post['Post']['id']; ?></td>
        <td>
            <?php echo $this->Html->link($post['Post']['title'],
array('action' => 'view', $post['Post']['id']));?>
        </td>
        <td>
            <?php echo $this->Form->postLink(
                'Delete',
                array('action' => 'delete', $post['Post']['id']),
                array('confirm' => 'Are you sure?'));
            ?>
            <?php echo $this->Html->link('Edit', array('action' => 'edit',
$post['Post']['id']));?>
        </td>
        <td>
            <?php echo $post['Post']['created']; ?>
        </td>
    </tr>
    <?php endforeach; ?>
</table>
```

Om vi nu surfar till vår index sida kommer den se ut på följande sätt:



Id	Title	Actions
9	vilken user	Delete Edit
10	mania	Delete Edit
11	kaja	Delete Edit
12	misiso	Delete Edit
13	ble ble	Delete Edit

Nu är vi klara!