



Linux  
Kullanıcıları  
Derneği



# CakePHP 2.x

Burak USGURLU  
[burak@uskur.com.tr](mailto:burak@uskur.com.tr)

# Tanışalım...



- Burak USGURLU
  - 348 no'lu üye.
  - 2007 Başkent Üniversitesi, Bilgisayar Mühendisliği
  - 2010 ODTÜ, Yazılım Yönetimi
  - Şu anda Uskur Yazılım'da

# Tanışalım...



- Sizler?
  - CakePHP duyanlar/ile geliştirenler?
  - Diğer PHP çerçevelerini (*framework*) kullananlar?
  - PHP olmayan çerçeve kullananlar?
  - Üye yazılımı çalışmalarını duymuş olanlar?

# Önümüzdeki 7 Gün



**2012-07-01**

CakePHP kuralları, bileşenleri (model, view, controller, behaviour, component, helper, ...)

**2012-07-02**

Uygulama Geliştirme: Uygulamaya genel bakış, veritabanının tasarımı, temel SVN kullanımı, iş paketleri.

2012-07-03 Üye Yazılımı Uygulama Geliştirme

2012-07-04 Üye Yazılımı Uygulama Geliştirme

2012-07-05 Üye Yazılımı Uygulama Geliştirme

2012-07-06 Üye Yazılımı Uygulama Geliştirme

2012-07-07 Üye Yazılımı Uygulama Geliştirme

2012-07-08 Ayrılış

# 7 Gün Sonunda...



## Amacımız:

- CakePHP kullanarak uygulama geliştirebiliyor olmanız.
- Çalışan bir üye yazılımımızın olması.

# Eğitim Akışı



- CakePHP Nedir?
- Neden? İyi ve Kötü Yanları
- CakePHP Kaynakları
- Model–View–Controller (MVC)
- CakePHP ile İstek Karşılama
- Cake: İçindekiler
- CakePHP Kuralları
- Denetçi (Controller)
- Bileşen (Component)
- Model
- Davranış (Behaviour)
- Görünüm (View)
- CakePHP Komut Satırı
- CakePHP'nin Gereksinimleri
- CakePHP Kuralım

# CakePHP Nedir?



- CakePHP, PHP için özgür ve açık kaynaklı (*MIT*) uygulama geliştirme çerçevesidir (*framework*).
  - Cake'in Ruby on Rails'ı örnek aldığı sıkça telafuz edilmektedir.
  - Symphony, CodeIgniter, Zend, Prado diğer PHP çerçevelerine örneklerdir.
- Programcılar için web uygulamaları geliştirmek için temel bir yapıdır.
- Size, uygulama geliştirirken esas yapmanız gereken kodlamayı, yani uygulamaya özel mantığı kodlamaya başlamak için ihtiyacınız olan tüm araçları sunar.

# Neden? İyi ve Kötü Yanları



- İyi
  - Hızlı geliştirilebilme
  - Güvenliğiniz başkalarının elinde
  - Tekerleği tekrar icat etmeniz gerekmiyor
  - Kodunuz belli bir standarda uyuyor
- Kötü
  - Performans
  - Güvenliğiniz başkalarının elinde
  - Hatalardan etkilenirsiniz
  - Yazılımınızı güncellemeniz gerekebilir



# CakePHP Kaynakları



- <http://cakephp.org> – Cake'in Anasayfası
- <http://book.cakephp.org> – Cake'in Yemek Kitabı (Kılavuzu)
- <http://api.cakephp.org> – Cake'in API'si
- <http://bakery.cakephp.org> – Fırın, CakePHP üzerine paylaşım alanı
- <http://groups.google.com/group/cake-php> – Topluluğun yardım ve paylaşım alanı
- [#cakephp](irc://irc.freenode.net) – Anında destek için IRC kanalı

# Sürümler



- İlk geliştirilmesine 2005 yılında başlandı.
- 1, 1.1 ve 1.2 sürümlerinin ardından...
- 1.3 sürümü 24 Nisan 2010
- 2.0 sürümü 16 Ekim 2011
- 2.1 sürümü 4 Mart 2012
- 2.2RC2 sürümü 19 Haziran 2012

1.x sürümleri arasında geçiş API'de gerçekleşen değişiklikler nedeniyle zorluydu.

2.x sürümleri arasında geçiş API'da değişiklikler olmadığı için nerdeyse sorunsuz.

Biz 2.2RC2'yi takip ediyor olacağız.

# CakePHP 2.x Gereksinimleri



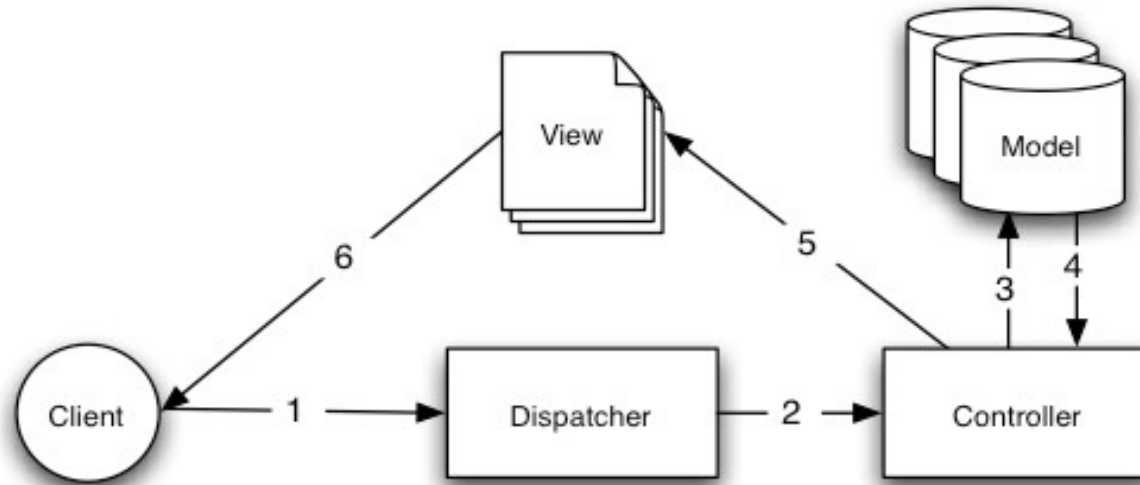
- HTTP Sunucusu. mod\_rewrite özelliği açık bir Apache sunucusu tercih edilir, ancak zorunlu değildir.
- PHP 5.2.8 veya daha yeni bir sürümü.
- Teknik olarak bir veritabanına ihtiyacınız yok ancak aşağıdakilerden herhangi birisini kullanabilirsiniz...
  - MySQL (4 veya daha yenisi), PostgreSQL, Firebird DB2, Microsoft SQL Server, SQLite, ...

# Model-View-Controller (MVC)



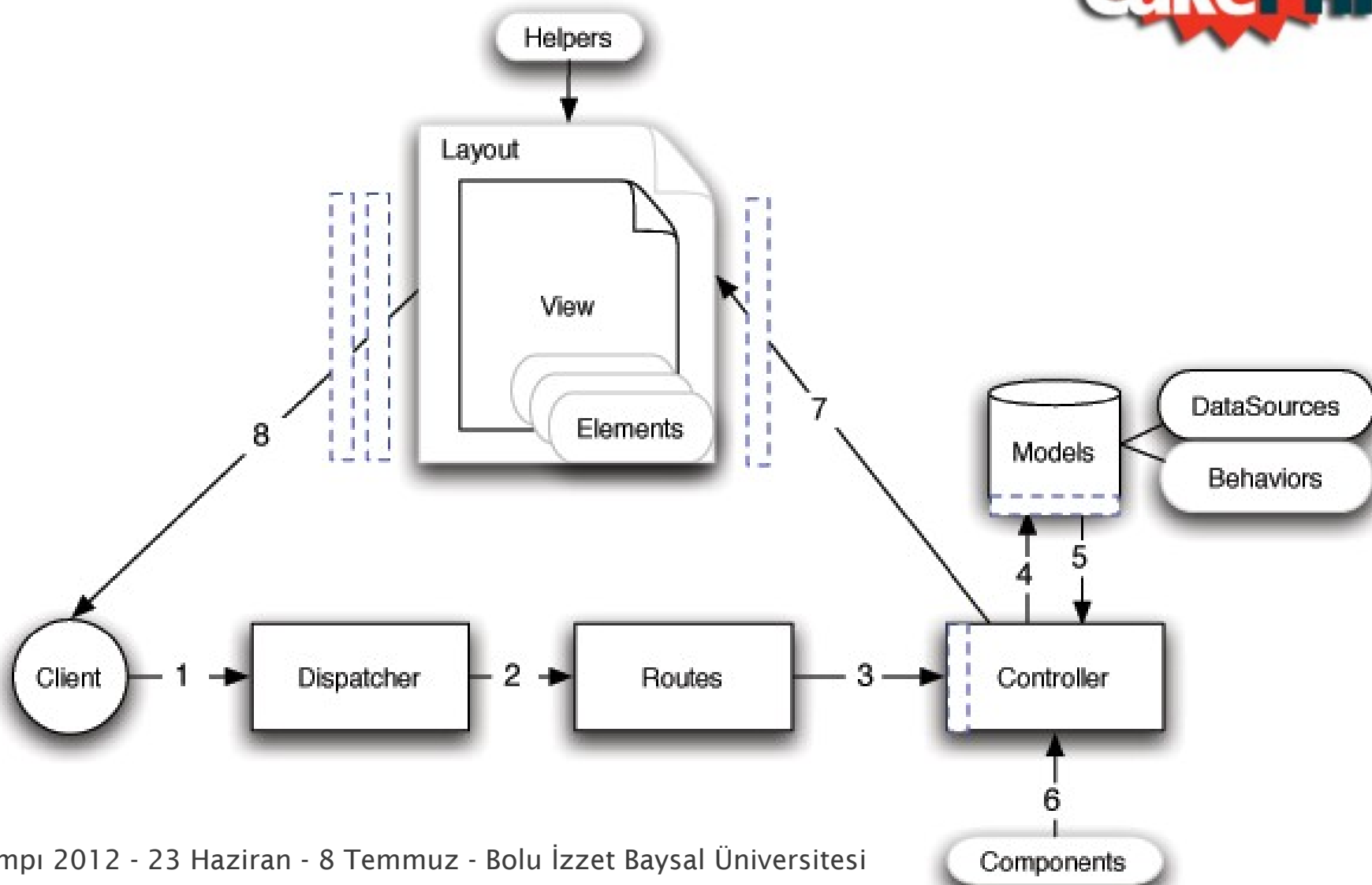
- Model-Görünüm-Denetçi (MGD)
- Bir yazılım tasarım deseni ve CakePHP bu deseni temel alır.
- Amaç iş mantığı ile uygulamanın çıktısını birbirinden ayırmaktır.
- Böylece bir katmanda oluşacak bir değişiklikten diğer katmandaki kodun etkilenmesi en aza indirilmiş olur.

# Model-View-Controller (MVC)



- Model kısmı uygulama verisini temsil eder
- Görünüm (View) kısmı model verisinin sunuşunu oluşturur
- Denetçi (Controller) istemciden gelen istekleri ele alır ve yönlendirir

# CakePHP ile İstek Karşılama



# Cake: İçindekiler

- CakePHP Denetçi (**Controller**), **Model** ve Görünüm (**View**) sınıflarına yer verir.
- Ayrıca Bileşenler (**Components**), Davranışlar (**Behaviors**) ve Yardımcı (**Helpers**) sınıfları ile genişleme ve tekrar kullanma olanakları sunar.
- **app**: Bizim uygulamamız.
- **lib**: CakePHP kütüphanesi barınır.
- **vendors**: Harici sınıflar ve kütüphaneler

- app
  - Config
  - Controller
  - Lib
  - Locale
  - Model
  - Plugin
  - tmp
  - Vendor
  - View
  - webroot
- lib
- vendors
- plugins
- .htaccess
- index.php
- README

# CakePHP Kuralları



- Dosya, sınıf, tablo gibi unsurları isimlendirme kuralları var.
- Sadece kurallara uyarak başka bir çaba harcamadan uygulamanıza özellik kazandırabilirsiniz.
- Kuralların kullanımı ayrı bir tanım dosyası tutma ihtiyacı ortadan kaldırmakta, ortak bir “dil” oluşturarak farklı geliştiricilerin uygulamayı kolayca anlamasını sağlamaktadır.



# Kurallar: İsimlendirme



Sınıf Türü	Sınıf İsmi	Dosya İsmi
Controller (Denetçi)	KucaklasmaVeOpucuklerController	KucaklasmaVeOpucuklerController.php
Bileşen (Component)	BenimKullanisliComponent	BenimKullanisliComponent.php
Model	SecenekDegeri	SecenekDegeri.php
Davranış (Behavior)	CokEsnekBehavior	CokEsnekBehavior.php
Görünüm (View)	AcayipBasitView	AcayipBasitView.php
Yardımcı (Helper)	SimdiyeKadarkiEnlyiHelper	SimdiyeKadarkiEnlyiHelper.php

- Aslında oldukça basit: (model hariç) sınıf ismi DeveSirti[SınıfTürü] biçiminde dosya isminde sınıf isminin sonunda .php var.

# Kurallar: Model ve Veritabanı

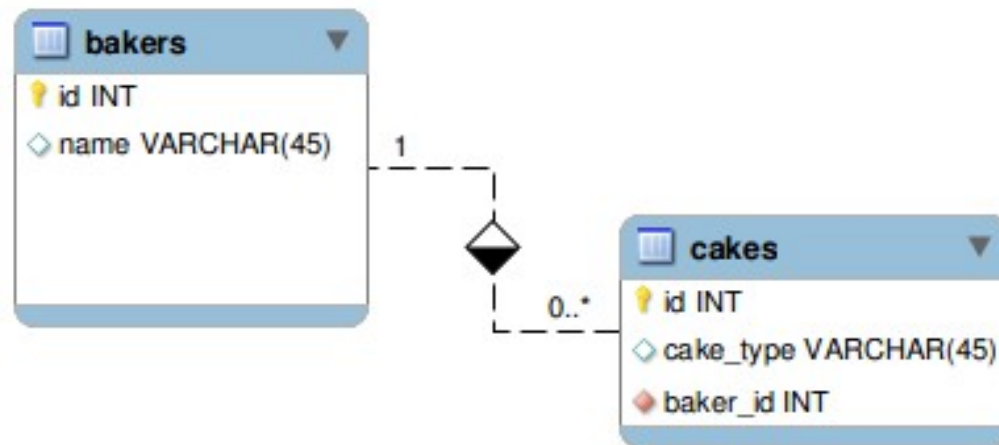


- **Model sınıf isimleri İngilizce, tekil ve DeveSirti biçiminde** olmalıdır. Person, BigPerson, ReallyBigPerson gibi.
- Modellerle ilişkili tablo **isimleri İngilizce, çoğul ve alttan çizgili** olmalıdır. Örneğin people, big\_people, really\_big\_people gibi.
- hasMany, belongsTo veyahasOne ilişkilerindeki yabancı anahtarlar (foreign keys) ön tanımlı olarak **ilişkili olduğu modelin (tekil) ismine \_id** eklenmesi ile tanımlanır. baker hasMany cakes ilişkisinde cakes tablosunda baker\_id yabancı anahtar olmalıdır.
- hasAndBelongsToMany (HABTM) ilişkilerinde, birleşim modeli **birleştirdiği modellerin isimlerinin alfabetik sırayla eklenmesi** ile isimlendirilmelidir.
- Tüm tablolarda her satırı tanımlamak için **tekil bir birincil anahtar (primary key) bulunmalıdır.**

# Örnek: Model ve Veritabanı İsim



- Model: Baker
- Tablo: bakers
- İlişki: Baker hasMany Cakes  $\Leftrightarrow$  Cake belongsTo Baker
- Yabancı Anahtar (Foreign Key): bakers  $\rightarrow$  baker\_id



# Kurallar: Denetçi



- **Denetçi sınıf isimleri İngilizce, çoğul ve DeveSirti** biçiminde olmalıdır. İsmin sonuna Controller eki eklenmelidir. PeopleController, BigPeopleController, ReallyBigPeopleController gibi.
- **URL'de denetçiler alttan çizgili** biçimde controller yazılmadan çağırılır. people, big\_people gibi.
- Bir denetçide oluşturulması **beklenen ilk eylem index()**'dir. Öntanımlı olarak `http://example.com/apples` çağırıldığında apples denetçisinin index() eylemi çağırılacaktır.

# Kurallar: Örnek



- Kurallara uyulduğunda  
<http://example.com/people/> isteği için...
- Veritabanı tablosu: "people"
- Model sınıfı: "Person"
  - Konumu: /app/Models/Person.php
- Denetçi Sınıfı: "PeopleController"
  - Konumu: /app/Controllers/PeopleController.php
- Denetçi Eylemi: index()
- Görünüm: /app/Views/People/index.ctp

# Kurallar: Örnek 2



- Kurallara uyulduğunda  
`http://example.com/books/view/12` isteğinde...
- Veritabanı tablosu: "books"
- Model sınıfı: "Books"
  - Konumu: `/app/Models/Books.php`
- Denetçi Sınıfı: "BooksController"
  - Konumu: `/app/Controllers/BooksController.php`
- Denetçi Eylemi: `view($id = null)`, parametre olarak `$id=12` atanır.
- Görünüm: `/app/Views/Books/view.ctp`

# Denetçi (*Controller*)



- Denetçi (controller) uygulama mantığını yönetmek için kullanılmaktadır.
- Genelde herbir denetçi tek bir model ile ilişkidir.

*RecipesController -> Recipe (model)*

- İçlerinde Eylem (Action) olarak isimlendirilen metodlar barındırırlar.
- Gelen isteğe göre ilgili Eylem çalıştırılır ve mantığın sonucu ilgili Görünüme (View) iletilir.
- RecipesController->AppController->Controller sınıfından türetilir.

# Denetçi Geri Çağrılar (Callback)



- Denetçilerde Geri Çağrılar (callback) bulunmaktadır.
- Geri Çağrılar vasıtasıyla CakePHP'nin çekirdek akışının içersine mantık gömebilirsiniz.
- Denetçi'de kullanılabilen geri çağrılar:
  - **beforeFilter()** çağrısı denetçideki herhangi bir eylemden önce çalıştırılır
  - **beforeRender()** çağrısı denetçideki eylem bittikten sonra görünüm oluşturulmadan önce çalıştırılır
  - **afterFilter()** çağrısı denetçideki eylem bittikten ve görünüm oluşturulduktan sonra çalıştırılır.



# Örnek Denetçi (*Controller*)



```
1. <?php
2. App::uses('AppController', 'Controller');
3. # /app/Controllers/RecipesController.php
4. class RecipesController extends AppController {
5.     public function view($id) {
6.         //eylem mantığı buraya..
7.     }
8.     public function share($customer_id, $recipe_id) {
9.         //eylem mantığı buraya..
10.    }
11.    public function search($query) {
12.        //eylem mantığı buraya..
13.    }
14. }
15. ?>
```

# Bileşen (Component)



- Farklı denetçiler arasında mantığı paylaşmaya olanak sağlayan paketlerdir.
- Her denetçide ihtiyaç duyulan ortak özellikler bir bileşen paketi haline getirilebilir.
- CakePHP'de hazır gelen bazı bileşenler:
  - **Sessions** – Kullanıcı oturumlarını yönetmek. Oturumda bilgi saklamak, bilgi okumak.
  - **Cookies** – Kullanıcıda saklanan çerezleri yönetmek için metodlar içerir.
  - **Authentication** – Kullanıcı kimliği doğrulama ve doğrulanmış kimlik oturumu açmak için kullanılır.
  - **Request Handling** – İstemciden gelen farklı taleplerin neler olduğunu bulmak için kullanılır. (http, json, RSS, vb.)

# Örnek: Pagination



- PaginationComponent sayfa başına kabul edilebilir miktarlarda kayıt gösterilmesini sağlamaktadır.

```
<?php
class PostsController extends AppController {

    public $paginate = array(
        'fields' => array('Post.id', 'Post.created'),
        'limit' => 25,
        'order' => array(
            'Post.title' => 'asc'
        )
    );

    public function list_recipes() {
        $data = $this->paginate('Post');
        $this->set('data', $data);
    }
}
```

# Model



- Model veriyi temsil eder ve nesne yönelimli programlamadaki “şey”e denk gelir. (ev, araba, kitap, yazar)
- Bir model başka modellerle ilişkilendirilebilir. Örneğin bir **Tarif**, o tarifin **Yazarı** ve tarifin **İçindeki Malzemeler** ile ilişkilendirilebilir.
- Modeller CakePHP'de veriye erişim için kullanılmaktadırlar.
- Recipe → AppModel → Model sınıfından türetilir.
- CakePHP veriyi kaydetmek, silmek ve aramak için **metodlar sunar.**

# Örnek Model



```
1.<?php
2.# /app/Models/Ingredient.php
3.class Ingredient extends AppModel {
4.    public $validate = array();
5.    public $belongsTo = array();
6.    public $hasMany = array();
7.
8.}
9.?>
```

# Modeller Arası İlişkilendirme



- CakePHP'de dört türde ilişki vardır.

İlişki	Bağ Türü	Örnek
bire bir	hasOne	Bir kullanıcının bir profili vardır.
one to one		
birden çoğa	hasMany	Sistemdeki kullanıcıların birden çok tarifi vardır.
one to many		
çoktan bire	belongsTo	Tarifler kullanıcıya aittir.
many to one		
çoktan çok	hasAndBelongsToMany	Tariflerin birden çok etiketi vardır ve birden çok etikete aittir.
many to many		

# Model - hasOne



- “Şey”ler arasındaki birebir ilişkiyi tanımlar.
- User has one Profile
  1. `<?php`
  2. `class User extends AppModel {`
  3. `var $hasOne = 'Profile';`
  4. `}`
  5. `?>`



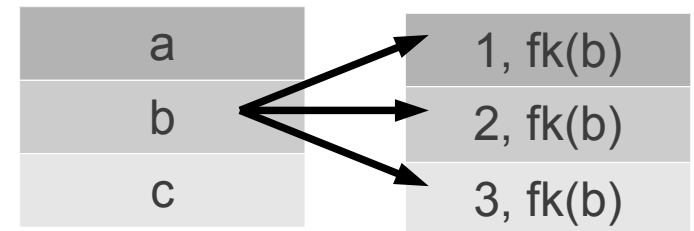
# Model - hasMany



- Bir “şey” ile sahip olduğu birden fazla “şey” arasındaki ilişkiyi tanımlar.

- User has many Comment(s)

```
1. <?php
2. class User extends AppModel {
3.     var $hasMany = array(
4.         'Comment' => array(
5.             'className' => 'Comment',
6.             'foreignKey' => 'user_id',
7.             'conditions' => array('Comment.status' => '1'),
8.             'order' => 'Comment.created DESC',
9.             'limit' => '5',
10.            'dependent'=> true
11.        )
12.    );
13. }
14. ?>
```



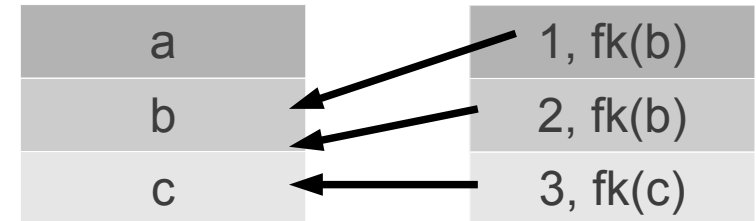


# Model - belongsTo



- Bir “şey” ile ait olduğu “şey”(ler) arasındaki ilişkiyi tanımlar.
- Eğer bir model yabancı anahtar (Foreign Key) içeriyor ise başka bir modele aittir.
- Profile belongs to User

```
1. <?php
2. class Profile extends AppModel {
3.     var $belongsTo = array(
4.         'User' => array(
5.             'className' => 'User',
6.             'foreignKey' => 'user_id'
7.         )
8.     );
9. }
10. ?>
```



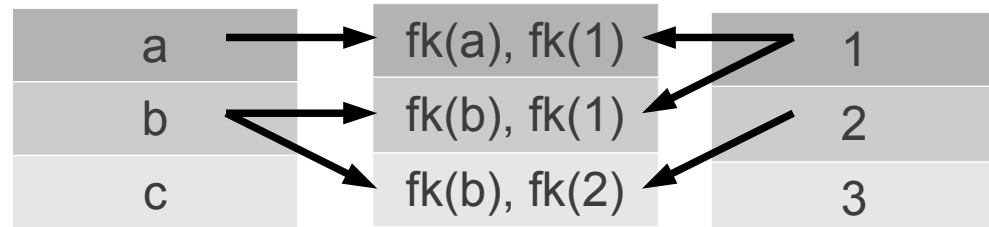
# Model - hasAndBelongsToMany



- Bir “şey”in başka “şey”lere sahip olduğu aynı zamanda başka “şey”lere ait olduğu ilişkiyi tanımlar.

- Recipe(s) HABTM Tag(s)

```
1. <?php
2. class Recipe extends AppModel {
3.     var $hasAndBelongsToMany = array(
4.         'Tag' =>
5.             array(
6.                 'className' => 'Tag',
7.                 'joinTable' => 'recipes_tags',
8.                 'foreignKey' => 'recipe_id',
9.                 'associationForeignKey' => 'tag_id',
10.                'unique' => true
11.            )
12.    );
13. }
14. ?>
```



# Model Geri Çağrılar (callbacks)



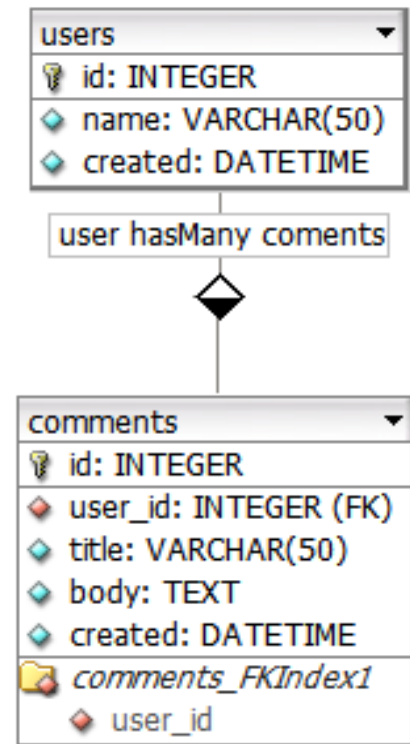
- Denetçilerde olduğu gibi Model'lerin de geri çağrı metodları bulunmaktadır.
  - **beforeFind()** çağrısı arama ile ilişkili bir metod çalıştırılmadan önce çağırılır.
  - **afterFind()** arama ile ilişkili bir metod çalıştırıldıktan sonra çağırılır.
  - **beforeValidate()** model içi doğrulama işlemi yapılmadan önce çağırılır.
  - **beforeSave()** verinin doğrulanmasından sonra veritabanına kayıt edilmeden hemen önce çağırılır.
  - **afterSave()** veri veritabanına yazıldıktan sonra çağırılır.
  - **before/afterDelete()** veri silinmeden önce/sonra çağırılır.
  - **onError()** eğer model katmanında bir hata oluşursa çağırılır.

# Model – hasMany örnek veri



- `$this->User->find()` çağrısı için örnek bir sonuç

```
Array
(
    [User] => Array
        (
            [id] => 121
            [name] => Gwoo the Kungwoo
            [created] => 2007-05-01 10:31:01
        )
    [Comment] => Array
        (
            [0] => Array
                (
                    [id] => 123
                    [user_id] => 121
                    [title] => On Gwoo the Kungwoo
                    [body] => The Kungwooness is not so Gwooish
                    [created] => 2006-05-01 10:31:01
                )
            [1] => Array
                (
                    [id] => 124
                    [user_id] => 121
                    [title] => More on Gwoo
                    [body] => But what of the 'Nut?
                    [created] => 2006-05-01 10:41:01
                )
        )
)
```



# Temel Model Metodları



- create() modeli yeni bir kayıt yazmak üzere sıfırlar.
- save(\$data) uygun biçimde aktarılmış veriyi veritabanına yazar.
- find() modelden sorgu yapar, sorgu sonucunu uygun biçimde döndürür.
- Denetçi içinden basit bir örnek:

```
1. $data['Person']['name']='Ahmet';
2. $data['Person']['last_name']='Güzelinsan';
3. $this->Person->create();
4. $this->Person->save($data);
5.
6. $result=$this->Person->find('all');
7. print_r($result);
8. Array
9. (
10.     [Person] => Array(
11.         [0] => Array(
12.             [name] => Ahmet
13.             [last_name] => Güzelinsan
14.         )
15.     )
16. )
```

# Model: Gelişmiş Sorgulama



```
$results=$this->Person->find('all',array(
    'conditions' => array('Person.gender' => 'm'),
    'recursive' => 1, //ne kadar derine
    'fields' => array('Person.name','Person.gender), //getirilecek alanlar
    'order' => 'Person.name ASC', //sıralama
    'limit' => 10, //kaç tane
));
```

Cinsiyeti erkek olan kişilerin ad ve cinsiyetlerini, adlarına göre sıralama yaptıktan sonra ilk 10 tanesini getir.

# Doğrulama (Validation)



- Veritabanına gönderilen verinin **kayıt edilmeden veya güncellenmeden önce doğrulanmasını** sağlar.
- Doğrulamanın nasıl yapılacağı ilgili **model sınıfı içerisinde \$validate** özeliğinin atanmasıyla belirlenir.
- CakePHP'nin kendi kural tanımları bulunmaktadır.
  - **alphaNumeric** //sadece sayı ve metine izin verir
  - **cc** // geçerli bir kredi kartı numarasımı?
  - **email** // geçerli bir epostamı?
  - **ip** //geçerli bir IP adresimi?
  - **isUnique** // bu alan tekil mi?
  - **url** //geçerli bir url mi?
  - ...
- Bunların dışında kendimiz fonksiyon yazarak başka doğrulama kuralları oluşturabiliriz.
- Her bir alan için bir kural belirlenebileceği gibi birden fazla kuralda tanımlanabilir.
- Kurallar aşağıdaki parametrelerle de güncellenebilir.
  - **Required=>>true** Kayıt edilmesi için bu alanın bulunması gerekiyor.
  - **AllowEmpty=>>true** Boş gönderilmesine izin veriliyor.
  - **On=>[create|update]** Kuralı yeni kayıt/kayıt güncelleme sırasında işlet.

# Örnek: Doğrulama



```
<?php
class User extends AppModel {
    public $validate = array(
        'alias' => array(
            'alphaNumeric' => array(
                'rule'      => 'alphaNumeric',
                'required' => true,
                'message'   => 'Alphabets and numbers only'
            ),
            'between' => array(
                'rule'      => array('between', 5, 15),
                'message' => 'Between 5 to 15 characters'
            )
        ),
        'password' => array(
            'rule'      => array('minLength', '8'),
            'message' => 'Minimum 8 characters long'
        ),
        'email' => 'email'
    );
}
```



# Davranış (Behaviour)



- Davranışlar doğrudan model ile alakalı olmayan ancak eklenmesi istenen özellikleri kolayca eklemeye olanak tanır.
- Örneğin bir modelimizde ağaç veri yapısını saklamak için veritabanını kullanmaya ihtiyaç duyalım.
- Sadece bu model için ağaç yapısına müdahale edecek metodlar yazmak yerine modelimize TreeBehaviour, ağaç davranışını ekleyerek modelin içinde tuttuğu veriye ağaç veri yapısı biçiminde davranması sağlanabilir.
- CakePHP ile pekçok hazır davranış gelmektedir. (ACL, Containable, Translate, Tree)

# Görünüm (*View*)



- Kullanıcıya konuştuğunuz katman.
- Görünüm dosyaları (.ctp uzantılı) düz PHP olarak yazılırlar ve  
`/app/Views/<controller>/<action>.ctp` konumuna yerleştirilirler.
- Örneğin Products denetçisinin `view()` eyleminin görünüm dosyası  
`/app/Views/Products/view.ctp`'dir.

# Görünüm Parçaları



- Görünüm dosyaları birkaç farklı parçadan oluşabilirler.
  - **layouts (yerleşim)** görünüm dosyalarını saran üst katman, görünüm dosyaları bu yerleşim dosyalarının içerisinde oluşturulurlar
  - **elements (öğeler)** farklı sayfalarda yerleşimin farklı yerlerinde tekrar edilmesi gereken öğeler “/app/Views/Elements/” dizinine yerleştirilerek farklı görünümlerden çağırılabilirler.
  - **helpers (yardımcılar)** farklı görünümde kullanılması gereken, görünümü işleyecek mantıkların tutulduğu sınıflardır. Yardımcılar vasıtasıyla örneğin formlar ve HTML kodları oluşturabilirsiniz.

# Örnek Yardımcı: HTML



- HtmlHelper HTML ile ilgili seçenekleri basit, hızlı değişikliğe daha kolay ayak uyduracak biçimde üretmek.

```
<?php //anchor tag
echo $this->Html->link(
    'Dan Brown Kitapları',
    array('controller'=>'books', 'action'=>'list', 'author_id'=>'2'),
    array('target'=>'_blank')
); ?>
<a href="/books/list/author_id:2" target="_blank">Dan Brown Kitapları</a>

<?php //image tag
echo $this->Html->image("kediler.jpg", array("alt" => "Kediler"))
?>


<?php
echo $this->Html->script(array('jquery', 'wysiwyg', 'scripts'));
?>
<script type="text/javascript" href="/js/jquery.js"></script>
<script type="text/javascript" href="/js/wysiwyg.js"></script>
<script type="text/javascript" href="/js/scripts.js"></script>
```

# Örnek Yardımcı: Session



- SessionComponent \$\_SESSION değişkeni üzerinden çalışarak birkaç kolaylık sağlar.
- SessionHelper, SessionComponent'un sahip olduğu özellikleri taklit ederek ona eşlik eder ve view içersinden session değerlerini okuma desteği verir.

//controller içersinde SessionComponent

```
<?php
$this->Session->write('Person.gender', 'male');
$this->Session->write('Person.city', 'ankara');
```

//view içersinde SessionHelper


```
<?php
$sessionData = $this->Session->read('Person');
print_r($sessionData);
'Person'=>Array(
    'gender'=>'male',
    'city'=>'ankara'
)
```

//controller içersinden

```
$this->Session->delete('Person.gender'); //gender değerini siler
$this->Session->destroy(); // session'ı sıfırlar
```

# Bir Görünüm (View) Örneği



 CakePHP: the rapid development php framework

## Add Comment

User

Ahmet Güzelinsan

Title

Body

Submit

[List Comments](#)

```
<div  
<?php  
<file
```

```
<?php  
echo  
echo  
echo  
?>
```

```
</file
```

```
<?php
```

```
</div
```

```
<div
```

```
<ul>
```

```
<li>
```

```
echo
```

```
array
```

```
//html
```

```
?>
```

```
</li>
```

```
</ul>
```

```
</div
```

```
Linux
```

# CakePHP Komut Satırı



```
$ cd /benim/cake/uygulamam  
$ ../cake/console/cake
```

- Komutları ile CakePHP komut satırını çağırabilirsiniz.
- Komut satırı vasıtasıyla,
  - Projenizi “pişirmeye” başlayabilirsiniz.
  - Otomatik kod oluşturma özelliğinden faydalanabilirsiniz.
  - Uygulamanızdaki bazı görevleri komut satırından yürütülecek biçimde ayarlayabilirsiniz. (cron)

# CakePHP Kuralım



1. Gereksinimler karşılanmış olmalı...
2. CakePHP'yi edinin. (arşiv paketleri, svn)
3. Web sunucunuzun bir dizinine paket içeriğini yerleştirin. (ör.: /var/www/html/cake2)
4. app/tmp dizinini web sunucusu yazabilecek biçimde ayarlayın.
5. `http://localhost/cake2` adresine gittiğimizde Cake'in karşılama ekranı ile karşılaşacağız...





# Teorik Kısımın Sonu

- Buraya kadar dayandığınız için teşekkür ederim...

Şimdi kek pişirebiliriz!