eBook Gratuit

APPRENEZ cakephp

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#cakephp

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: cakephp

It is an unofficial and free cakephp ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official cakephp.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec cakephp

Remarques

Cette section fournit une vue d'ensemble de ce qu'est cakephp et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans cakephp, et établir un lien avec les sujets connexes. La documentation de cakephp étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
1.2.0	2008-12-26
1.3.0	2010-04-25
2.0.0	2011-10-17
3.0.0	2015-03-22

Examples

Installation ou configuration

Exigences

Le guide d'installation suivant est destiné à cakephp 2.8 et versions ultérieures. Toutes les versions de cakephp inférieures à 2.8 ne sont pas compatibles avec php 7

Serveur HTTP. Par exemple: Apache. Avoir mod_rewrite est préférable, mais en aucun cas requis.

- PHP 5.5.9 ou supérieur (y compris PHP 7).
- extension PHP mbstring
- Intl extension PHP

  Attention! Dans XAMPP et WAMP, l'extension mbstring fonctionne par défaut. Dans XAMPP, l'extension intl est incluse, mais vous devez décommenter extension = php_intl.dll dans php.ini et redémarrer le serveur via le panneau de configuration XAMPP. Dans WAMP, l'extension intl est "activée" par défaut mais ne fonctionne pas. Pour que cela fonctionne, vous devez aller dans le dossier php (par défaut) C: \ wamp \

bin \ php \ php {version}, copiez tous les fichiers qui ressemblent à icu * .dll et collez-les dans le répertoire apache bin C: \ wamp \ bin \ apache \ apache {version} \ bin. Puis redémarrez tous les services et ça devrait aller.

Bien qu'un moteur de base de données ne soit pas requis, nous pensons que la plupart des applications en utiliseront un. CakePHP prend en charge une variété de moteurs de stockage de base de données:

- MySQL (5.1.10 ou supérieur)
- PostgreSQL
- Microsoft SQL Server (2008 ou supérieur)
- SQLite 3

Structure de dossier CakePHP3

Après avoir téléchargé, voici les fichiers et les dossiers que vous devriez voir:

- Le **dossier bin** contient les exécutables de la console Cake.
- Le **dossier de configuration** contient les fichiers de configuration utilisés par CakePHP. Les détails de connexion à la base de données, l'amorçage, les fichiers de configuration de base et plus doivent être stockés ici.
- Le **dossier plugins** est l'endroit où les plugins utilisés par votre application sont stockés.
- Le **dossier des journaux** contient normalement vos fichiers journaux, en fonction de votre configuration de journal.
- Le **dossier src** sera l'endroit où les fichiers de votre application seront placés.
- Le **dossier de tests** sera l'endroit où vous placez les cas de test pour votre application.
- Le **dossier tmp** est l'endroit où CakePHP stocke les données temporaires. Les données réelles stockées dépendent de la configuration de CakePHP, mais ce dossier est généralement utilisé pour stocker des descriptions de modèles et parfois des informations de session.
- Le **dossier du fournisseur** est l'endroit où CakePHP et les autres dépendances d'application seront installés. S'engager personnellement à ne pas modifier les fichiers de ce dossier.
- Le **répertoire webroot** est la racine du document public de votre application. Il contient tous les fichiers que vous voulez être accessibles publiquement.

  Assurez-vous que les dossiers **tmp** et **logs** existent et sont accessibles en écriture, sinon les performances de votre application seront sérieusement affectées. En mode débogage, CakePHP vous avertira si ce n’est pas le cas.

Dossier src intérieur

Le dossier src de CakePHP est l'endroit où vous réaliserez l'essentiel du développement de vos applications.

Le dossier de la console contient les commandes de la console et les tâches de la console pour votre application. Pour plus d'informations, voir Coques, tâches et outils de la console.

Le dossier du **contrôleur** contient les contrôleurs de votre application et leurs composants.

Le dossier **Locale** stocke les fichiers de chaîne pour l'internationalisation.

Le dossier **modèle** contient les tables, les entités et les comportements de votre application.

View - les classes de présentation sont placées ici: cellules, helpers et fichiers de modèle.

Template - les fichiers de présentation sont placés ici: éléments, pages d'erreur, mises en page et fichiers de modèles de vue.

Premier projet de base vide

Création initiale et téléchargement (CakePHP 3.x)

La manière la plus simple de créer un projet CakePHP est via Composer (si vous ne connaissez pas le compositeur, regardez ici pour plus d'informations)

Installez Composer

Si vous devez l'installer et que vous êtes sur une machine Windows, suivez ce guide

Si vous utilisez Linux / Unix / OSX, suivez ce guide

Créer le premier projet CakePHP

Ouvrez une fenêtre de console et accédez à votre installation de php (sous Windows avec l'installation par défaut de xampp, il s'agit de `C:\xampp\php`)

Pour créer un projet vide, exécutez la commande suivante:

```
php composer.phar create-project --prefer-dist cakephp/app name_of_your_project
```

Cuisson / Modèle / Vue / Contrôleurs

La magie de CakePHP est la cuisson - une génération automatisée de contrôleur, de modèle et de code de vue avec des options de base CRUD.

Avant la cuisson, votre connexion à la base de données doit être configurée. Pour ce faire, vous devez éditer le fichier `config/app.php` dans votre projet.

```
'Datasources' => [
'default' => [
    'className' => 'Cake\Database\Connection',
    'driver' => 'Cake\Database\Driver\Mysql',
    'persistent' => false,
    'host' => 'localhost',
    'username' => 'my_app', //in basic xampp: root
    'password' => 'sekret', //in basic xampp: ''
```

```
    'database' => 'my_app', //name of the database you want to connect to your project
    'encoding' => 'utf8',
    'timezone' => 'UTC',
    'cacheMetadata' => true,
]
```

],

Si votre base de données est correctement connectée, entrez `bin/cake bake` dans le dossier racine de votre projet dans une fenêtre de la console.

Cela devrait produire quelque chose comme ceci:

```
Welcome to CakePHP v3.1.6 Console
---------------------------------------------------------------
App : src
Path: /var/www/cakephp.dev/src/
PHP: 5.5.8
---------------------------------------------------------------
The following commands can be used to generate skeleton code for your application.

Available bake commands:

- all
- behavior
- cell
- component
- controller
- fixture
- form
- helper
- mailer
- migration
- migration_snapshot
- model
- plugin
- shell
- shell-helper
- template
- test

By using `cake bake [name]` you can invoke a specific bake task.
```

Pour simplifier, nous allons tout faire avec les paramètres par défaut. Pour ce faire, vous entrez

```
cake bake all
```

Cela produira quelque chose dans ce sens:

```
Welcome to CakePHP v3.2.11 Console
---------------------------------------------------------------
App : src
Path: C:\xampp\htdocs\tipping\src\
PHP : 5.6.15
---------------------------------------------------------------
Bake All
```

```
---------------------------------------------------------------
Possible model names based on your database:
- users
- blogs
Run `cake bake all [name]` to generate skeleton files.
```

En exécutant `cake bake all <modelNameYouWantToBake>` les `cake bake all <modelNameYouWantToBake>` modèle, de table, de contrôleur, de fixture et de vue sont créés. Exécutez ceci pour chaque nom de modèle possible et vous avez un projet fonctionnel avec des options de base CRUD.

Maintenant, vous pouvez ouvrir votre navigateur et voir à quoi il ressemble et commencer à étendre le projet par votre propre logique

Exigences

```
1-HTTP Server. For example: Apache. Having mod_rewrite is preferred, but by no means required.
2-PHP 5.5.9 or greater (including PHP 7)
3-mbstring PHP extens ion
4-intl PHP extension
```

Je fais généralement une installation apache et mysql sur une linuxbox. Je peux aussi utiliser Windows, mais je ne le recommande pas;) Donc, je fais généralement une nouvelle entrée dans le fichier / etc / hosts pour rendre un sitename disponible pour cakephp.

```
127.0.0.1   localhost caketest.local
```

Etape suivante pour copier tous les fichiers CakePHP dans un sous-répertoire de / home / myusername / public_html / caketest

```
app
cake
index.php
plugins
README
vendors
.htaccess
```

puis j'ai installé le site sur apache (pas nécessaire),

<VirtualHost *: 80> DocumentRoot "/ home / myusername / public_html / caketest" NomServeur caketest.local
Cela doit être omis dans l'environnement de production. Développement SetEnv APPLICATION_ENV

```
<Directory "/home/myusername/public_html/caketest">
Options Indexes MultiViews FollowSymLinks
AllowOverride All
Order allow,deny
Allow from all
</Directory>
```

redémarrer apache. vous devez également éditer les fichiers .htaccess et placer une directive RewriteBase avec le chemin hte dans le répertoire réel, par exemple

```
RewriteBase /~myusername/caketest
```

créer une base de données, définir la connexion db dans les fichiers de configuration de gâteau et c'est tout. vous pouvez pointer votre navigateur vers http: //caketest.local si vous ne voulez pas d'URL de site de test que vous pouvez ignorer, et apache vhost creation, mais l'URL à utiliser doit être http: / localhost / ~ myusername / caketest

Une autre chose importante est d'activer userdir modul dans apache, et aussi de vérifier si l'utilisation de php est également activée dans userdirs.

CakePHP 2.x Introduction de base

Parlera de la structure de répertoire de CakePHP, ce que chaque dossier signifie.

CakePHP a des dossiers principaux

1. app - Il contient notre code source d'application, tout notre code se trouve sous ce répertoire.
2. lib - Ceci est le libraire de base de cakephp, il contient tout le code de base de la bibliothèque de cakephp. La modification du code à l'intérieur de ce répertoire n'est pas suggérée car ils peuvent provoquer des erreurs lors de la mise à niveau de la bibliothèque Cakephp.
3. plugins - Ceci contient le code des plugins de cakephp qui sera utilisé pour notre application.
4. vendeurs - Ceci contient du code externe, Ce code n'utilisera pas la bibliothèque Cakephp.
5. index.php - Ceci est le fichier d'index.

   Nous pouvons avoir plusieurs applications hébergées dans un même projet. c'est-à-dire qu'ils peuvent utiliser les mêmes dossiers, plug-ins et fournisseurs de lib.

   Pour modifier le code de la lib, il est recommandé de les étendre dans notre dossier d'application et d'effectuer les modifications.

   Les plug-ins et les dossiers des fournisseurs sont partagés par toutes les applications hébergées dans le même répertoire.

   index.php est le fichier appelé en premier.

Maintenant, nous devrions sauter dans notre dossier d'application

Lire Démarrer avec cakephp en ligne: https://riptutorial.com/fr/cakephp/topic/958/demarrer-avec-

cakephp

Chapitre 2: Conseils de codage CakePHP3

Examples

Créer un nouveau contrôleur

```
namespace App\Controller;

class PostsController extends AppController {

    public function initialize(){
        parent::initialize();
        // code that you want to run before every action
    }
    public function view($id) {
       //Your code here
    }
}
```

Ajouter la méthode beforeFilter () dans Controller

La méthode beforeFilter () s'exécute avant toute autre méthode exécutée dans le contrôleur.

Utilisez d'abord l'espace de noms Event avant de définir la classe dans votre fichier de contrôleur.

```
use Cake\Event\Event;
```

Dans votre contrôleur, ajoutez la méthode beforeFilter () comme indiqué ci-dessous.

```
public function beforeFilter(Event $event) {
    parent::beforeFilter($event);
}
```

Ou, vous pouvez utiliser la méthode `initialize()` .

```
public function initialize(){
    parent::initialize();
}
```

Passer des variables à la vue

Passer chaque variable à afficher à la fois

```
$this->set('color', 'pink');
$this->set('color', $color);
```

Passer plusieurs variables à visualiser ensemble via la fonction compact ()

```
$color1 = 'pink';
$color2 = 'red';
$this->set(compact('color1', 'color2'));
```

Récupération de données de post équivalentes à $ _POST

Vous pouvez récupérer des données de publication en tant que tableau.

```
$post_data= $this->request->data;
```

Vous pouvez récupérer des données post pour une clé particulière.

```
$this->request->data['field'];
```

Récupérer une valeur de clé spécifique

```
$this->request->data('key_name');
```

Récupérer une valeur de clé spécifique d'un tableau imbriqué

```
$this->request->data('data.subfield');
```

la différence entre la notation du tableau et `data()` méthode `data()` est que `data()` est sans erreur et renvoie `null` si la clé n'existe pas dans le tableau

tellement de faire

```
if(isset($this->request->data['field']) && $this->request->data['field']) { ...}
```

tu peux faire

```
if($this->request->data('field')) { ...}
```

pour CakePHP 3.4.x +

obtenir toutes les données:

```
$this->request->getData();
```

obtenir une clé spécifique:

```
$this->request->getData('key');
```

pour définir les données disponibles pour la fonction getData, vous devez faire quelque chose comme:

```
$this->request = $this->request->withData('some_key_on_the_fly', 'value');
```

```
$some_key_on_the_fly = $this->request->getData('some_key_on_the_fly');
```

utile pour mettre à jour les modèles dans le contrôleur avec des données statiques

Charger un autre modèle dans le contrôleur

Par défaut, CakePHP charge le modèle associé dans le contrôleur. Pour charger un autre modèle dans le contrôleur, utilisez la méthode loadModel ():

```
$this->loadModel('Articles');
```

ou charger à la volée

```
$table = TableRegistry::get('Articles');
$table->find();
```

Redirection vers une autre page du contrôleur

Rediriger dans l'application (une autre action du contrôleur spécifique).

```
return $this->redirect([
    'controller' => 'myController',
    'action' => 'myAction'
]);
```

Rediriger vers la page de référence

```
return $this->redirect($this->referer());
```

Rediriger en dehors de l'application ou de l'URL spécifique

```
return $this->redirect("http://stackoverflow.com/users/1793428/haresh-vidja");
```

Passage d'une variable à une action à partir d'une URL avec redirection

Passer une variable dans l'URL comme **paramètre d'** une **méthode**

```
return $this->redirect([
    'controller' => 'users',
    'action' => 'profile',
    $id
]);
```

L'URL devrait ressembler à ceci : http: // your_app_url / users / profile / {id}

dans le fichier UsersController.php de la méthode profile ()

```
class UsersController extends Controller {
```

```
    public function profile($id=null) {
        $userData=$this->Users->get($id);
    }
}
```

Passer une variable dans l'URL en tant que **chaîne de requête**

```
return $this->redirect([
    'controller' => 'users',
    'action' => 'profile',
    '?'=>['id'=>$id]
]);
```

L'URL devrait ressembler à ceci : http: // votre_app_url / users / profile /? Id = {id}

dans le fichier UsersController.php de la méthode profile ()

```
class UsersController extends Controller {
    public function profile() {
        $userData=$this->Users->get($this->request->query('id'));
    }
}
```

Définir ou modifier la disposition de l'application

Définissez la mise en page par défaut **pour l'ensemble de l'application** . c'est-à-dire, fichier de mise en page créé dans /src/Template/Layout/admin.ctp

```
class AppsController extends Controller {

    public function beforeFilter(Event $event) {
        parent::beforeFilter($event);
        $this->viewBuilder()->layout('admin'); // For Version >= 3.1 or
        $this->layout = 'admin'; // for version < 3.1

        // your other code should be here
    }
}
```

Définissez la mise en page par défaut **pour une action spécifique dans l'application** . c'est-à-dire que l'application a une disposition différente dans la page de connexion dans /src/Template/Layout/login.ctp

```
class UsersController extends Controller {

    public function login() {

        $this->viewBuilder()->layout('login'); // For Version >= 3.1 or
        $this->layout = 'login'; // for version < 3.1

        //your other code should be here
    }
}
```

Modifier la mise en page **pour un contrôleur spécifique** . par exemple, vous avez besoin d'une mise en page différente pour toute méthode de contrôleur spécifique

la classe UsersController étend Controller {

```
    public function beforeFilter(Event $event) {
        parent::beforeFilter($event);

        $this->viewBuilder()->layout('user_layout'); // For Version >= 3.1 or
        $this->layout = 'user_layout'; // for version < 3.1

        //your other code should be here
    }
}
```

Définir la disposition de la requête Ajax

Généralement dans AJAX demande pas besoin de charger CSS, JS. En omettant également d'autres codes HTML.

Créer le fichier ajax.ctp dans / src / Template / Layout, et le code devrait être

```
<?php
    $this->fetch('content');
```

Définir la mise en page basée sur AJAX pour l'application entière, dans AppsController.php

class AppsController étend Controller {

```
public function beforeFilter(Event $event) {
    parent::beforeFilter($event);
    if($this->request->isAjax())
    {
        $this->viewBuilder()->layout('ajax'); // For Version >= 3.1 or
        $this->layout = 'ajax'; // for version < 3.1

    }
    else
    {
        $this->viewBuilder()->layout('admin'); // For Version >= 3.1 or
        $this->layout = 'admin'; // for version < 3.1
    }

    // your other code should be here
}
```

}

Charger des composants dans CakePHP

Nous pouvons charger des composants de deux manières.

1. Par initialiser ou remplacer la propriété $ components dans Controller

2. En utilisant la méthode loadComponent () dans la méthode initialize () du contrôleur.

Way-1 Il devrait être surcharger le composant par AppsController.php charger un ou plusieurs composants

```
class UsersController extends AppController {
    public $components = ['RequestHandler','Auth','Flash'];
}
```

Way-2 Utilisez cette méthode lorsque vous avez besoin d'un composant de chargement dynamique pour un contrôleur spécifique. Charger un composant

```
class UsersController extends AppController {
    public function initialize() {
        parent::initialize();
        $this->loadComponent("RequestHandler"); // load specific component
        $this->loadComponent(["RequestHandler","Auth","Flash"]); // load specific component
    }
}
```

Quelle est la méthode initilaize ()?

initialize () est introduit dans la version de CakePHP> 3.0

En tant que structure de code, elle ressemble à la méthode beforeFilter (). mais il y a beaucoup de différences entre beforeFilter () et initialize ().

1. initialize () est toujours appelée après l'appel du constructeur. mais beforeFilter () n'appelle pas en cas de méthode d'action introuvable dans un contrôleur particulier.
2. La méthode initialize () est généralement utilisée pour initialiser quelque chose comme ajouter de nouveaux composants et assistants. mais dans beforeFilter () est généralement utilisé pour exécuter une partie logique globale.

Retracer des données de chaîne de requête équivalentes à $ _GET

Vous pouvez récupérer des données de chaîne de requête sous forme de tableau.

```
$post_data= $this->request->query;
```

Vous pouvez récupérer des données post pour une clé particulière.

```
$this->request->query['field'];
```

Récupérer une valeur de clé spécifique

```
$this->request->query('key_name');
```

Récupérer une valeur de clé spécifique d'un tableau imbriqué

```
$this->request->query('data.subfield');
```

Création d'une classe de table (modèle)

Comment créer une classe de modèle d'utilisateur

```
namespace App\Model\Table;
use Cake\ORM\Table;

class UsersTable extends Table {
    public function initialize(array $config) {
        $this->table('users'); //define table name
        $this->displayField('username'); // unique or other special field of users table
        $this->primaryKey('id'); // primary key of users table
        $this->tablePrefix('prefix_'); // if prefix set tablename should be prefix_users

        // your other code here
    }

    // your other methods here
}
```

Associations de mannequins à CakePHP

Il y a 4 types d'associations (relations) que nous pouvons définir dans CakePHP

```
class PostsTable extends Table {
    public function initialize(array $config) {

        // table initialization code should be here

        $this->belongsTo('Authors', [
            'className' => 'Authors',
            'foreignKey' => 'author_id',
            'joinType' => 'INNER',
        ]);
        $this->hasMany('Tags');
        $this->hasOne('Categories');
        $this->hasAndBelongsToMany('Topics');
    }
}
```

Dans l'exemple ci-dessus, vous pouvez voir 4 types de relations

Post **appartient à** Author **(One to One)** , cela signifie que la table des `posts` a une clé étrangère `author_id` associée à la table `id` of `authors` .

Post **a beaucoup de** balises **(One to Many)** , cela signifie `tags` table des `tags` contient une clé étrangère `post_id` associée à l' `id` de la table des `posts` .

Publier **a une** balise **(plusieurs à un ou un à un)** , cela signifie que la table des `posts` a une clé étrangère `category_id` associée à la table `id` de `categories` .

Post **a et appartient à** Topics **(Many to Many)** , il y a beaucoup à beaucoup de relations entre les

`posts` et la table des `topics` . pour entretenir plusieurs relations doivent avoir besoin de créer une troisième table, table, le nom de la table devrait être `posts_categories` . Champs de cette table comme mentionné ci-dessous

1. id (clé primaire de la table)
2. post_id (clé étrangère de la table des `posts`)
3. topic_id (clé étrangère de la table des `topics`)

Lire Conseils de codage CakePHP3 en ligne:
https://riptutorial.com/fr/cakephp/topic/3341/conseils-de-codage-cakephp3

Chapitre 3: Modèles d'instanciation à partir d'une autre source de données

Remarques

À un moment donné, votre application CakePHP devra interroger plusieurs bases de données. La méthode de demande de modèles à partir de bases de données autres que celles par défaut n'est pas présente dans la documentation officielle.

Examples

Instancier utilise App :: uses

```
App::uses('YourModel', 'Model');
$model_1 = new YourModel(array('ds' => 'default'));
$model_2 = new YourModel(array('ds' => 'database2'));
```

La base de données à la volée change pour modal

Pour plusieurs bases de données, vous disposez du fichier database.php dans lequel vous pouvez définir autant de bases de données que nécessaire.

Si vous souhaitez "basculer" une base de données pour un modèle spécifique à la volée, utilisez la méthode setDataSource () .

Par exemple, si vous avez deux bases de données, vous pouvez les définir dans le fichier database.php comme "default" et "sandbox", par exemple.

Ensuite, dans votre code:

$ this-> MyModal-> setDataSource ('sandbox');

Le bac à sable est le nom de la configuration et le nom réel de la base de données est écrit une seule fois dans le fichier database.php.

Lire Modèles d'instanciation à partir d'une autre source de données en ligne: https://riptutorial.com/fr/cakephp/topic/1953/modeles-d-instanciation-a-partir-d-une-autre-source-de-donnees

Chapitre 4: Traitement des requêtes Ajax

Examples

Exemple de base de CakePHP 2.x

Controller: Dans le Controller, vous devez ajouter le composant RequestHandler. Cela permet à CakePHP de détecter automatiquement les requêtes Ajax (voir: http://book.cakephp.org/2.0/en/core-libraries/components/request-handling.html pour plus d'informations):

```
class YourController extends AppController {
    public $components = array('RequestHandler');
    //...

    public function ajaxCall() {
        if($this->request->is('ajax')){
            // some code that should be executed
            // ...
            // variables you want to return
            $this->set(compact('firstVariable', 'secondVariable'));
            $this->set('_serialize', array('firstVariable', secondVariable))
    }
}
```

Voir le code (en utilisant jQuery):

```
<script>
$.ajax({
    type: 'POST',
    url: '/yourController/ajaxCall',
    success: function (result) {
        // result is a JSON array of the returned Variables
    },
    error: function (result){
        console.log(result);
    }
});
</script>
```

Requête Ajax dans Cakephp 2.x

Une autre façon d'utiliser ajax dans CakePHP. Cakephp se fournit pour une requête ajax. S'il vous plaît voir l'exemple.

```
 $data = $this->Js->get('#id')->serializeForm(array('isForm' => true, 'inline' => true));
//on click send request to controller and displays response data in view
$this->Js->get('#button-id')->event(
        'click', $this->Js->request(
                array('controller' => 'Users', 'action' => 'add'), array(
            'update' => '#time', // field you wish to update
```

```
            'data' => $data, // Form Data in serialize form
            'async' => true,
            'dataExpression'=>true,
            'method' => 'POST' // method get or post
                )
        )
);
```

Lire Traitement des requêtes Ajax en ligne: https://riptutorial.com/fr/cakephp/topic/6230/traitement-des-requetes-ajax

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec cakephp	burzum, Community, Deejay, gins t, Gorakh Yadav, Max90, mcgowan.b
2	Conseils de codage CakePHP3	arilia, CodeZilla, Haresh Vidja, IWillScoop, Max90, Miheretab Alemu, parsaya
3	Modèles d'instanciation à partir d'une autre source de données	Holt, RavatSinh Sisodiya, Zetaphor
4	Traitement des requêtes Ajax	Max90, Rashid Khan