PHP Framework

Phalcon :: A full-stack PHP framework delivered as a C-extension

Phalcon Framework

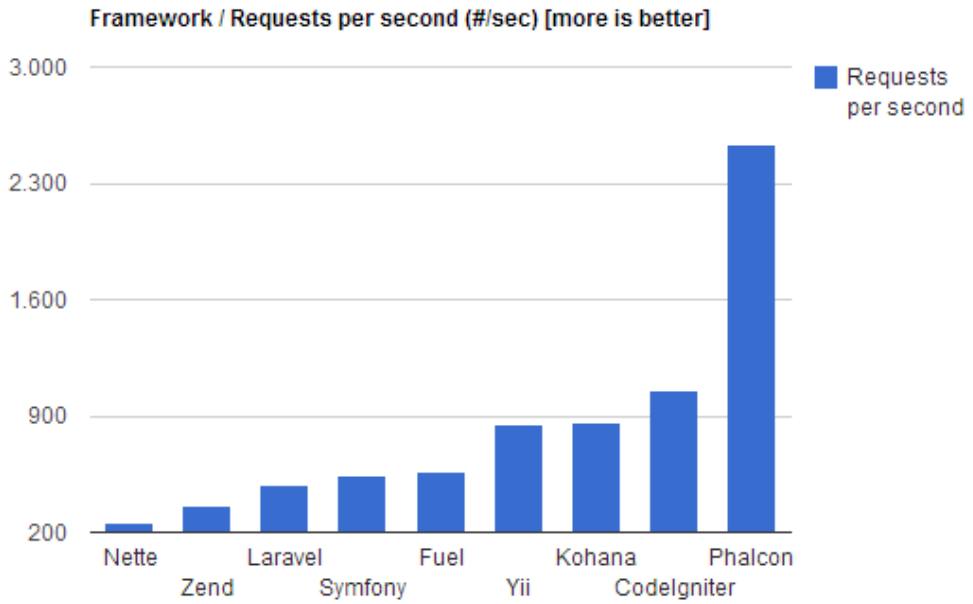
- เป็น full stack framework
- สถาปัตยกรรมการออกแบบเป็นแบบ MVC (Model-View-Control)
- รองรับ ORM (Object Relational Mapping)
- รองรับ ODM (Object Document Mapper)

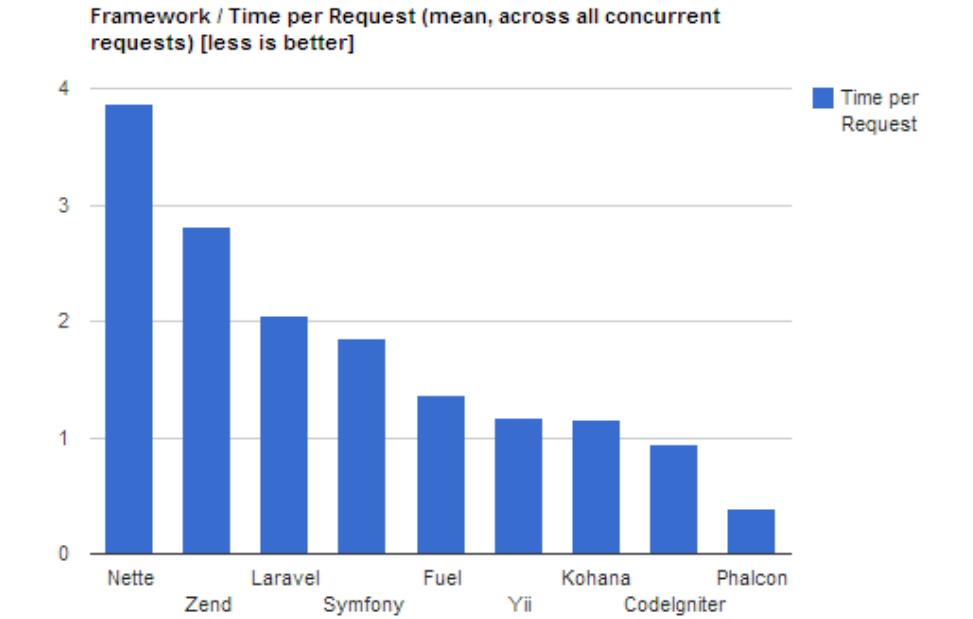
ดีกว่ายังไง?

phalcon จะอยู่ในรูปแบบของ PHP module (ไฟล์ .dll ใน windows หรือ .so ใน linux) phalcon จะถูกโหลดขึ้นมาในหน่วยความจำเมื่อเริ่มการทำงานของเว็บเซิร์ฟเวอร์ ในขณะที่เฟรมเวิร์คอื่นจะเป็นไฟล์ php ในรูปแบบ skeleton แล้วพัฒนาเพิ่มเติมตามโครงร่างนั้น ซึ่งในขั้นตอนการเรียกใช้จะต้องโหลดมาแปลด้วย php ทีละไฟล์ซึ่งทำให้ประสิทธิภาพด้อยกว่า

การทำงานของโปรแกรมที่เขียนด้วย PHP ขั้นตอนแรกจะต้องอ่านไฟล์โค้ด PHP มาแปลงเป็น bytecode แล้วจึงประมวลผล ยิ่งมีไฟล์จำนวนมาก ยิ่งทำให้การประมวลผลต้องใช้เวลานาน ซึ่งเป็นเหตุผลหลักที่ทำให้ PHP ด้อยกว่าภาษาอื่น เช่น Ruby หรือ Python ในแง่ของประสิทธิภาพ

Phalcon ลบข้อเสียนี้โดยการพัฒนาเฟรมเวิร์คด้วยภาษา C แล้วคอมไพล์เป็นโมดูลของ PHP ดังนั้นในการทำงานของแอพพลิเคชัน จึงอ่านข้อมูลเฉพาะส่วนที่พัฒนาขึ้นเท่านั้น


Framework / Requests per second (#/sec) [more is better]
3.000
2.300
1.600
900
200
Requests per second
Nette
Zend
Laravel
Symfony
Fuel
Yii
Kohana
CodeIgniter
Phalcon


Framework / Time per Request (mean, across all concurrent requests) [less is better]
4
3
2
1
0
Time per Request
Nette
Zend
Laravel
Symfony
Fuel
Yii
Kohana
CodeIgniter
Phalcon

BASIC FEATURES

- **Low overhead**
  Low memory consumption and CPU compared to traditional frameworks
- **MVC & HMVC**
  Modules, components, models, views and controllers
- **Dependency Injection**
  Dependency Injection and Location of services and it's itself a container for them.
- **Rest**
  In this case, you can use either a micro or full stack application to meet your goal. In addition, a powerful set of HTTP helpers.
- **Autoloader**
  Provides the autoloading mechanism of PHP classes following PSR-4.
- **Router**
  Phalcon\Mvc\Router provides advanced routing capabilities.

Low overhead

PHP C-extension (Phalcon)

- Zephir/C extensions are loaded together with PHP one time on the web server's daemon start process
- Classes and functions provided by the extension are ready to use for any application
- The code is compiled and isn't interpreted because it's already compiled to a specific platform and processor
- Thanks to its low-level architecture and optimizations **Phalcon provides the lowest overhead for MVC-based applications**

BASIC FEATURES

Low overhead
Low memory consumption and CPU compared to traditional frameworks

MVC & HMVC
Modules, components, models, views and controllers

Dependency Injection
Dependency Injection and Location of services and it's itself a container for them.

Rest
In this case, you can use either a micro or full stack application to meet your goal. In addition, a powerful set of HTTP helpers.

Autoloader
Provides the autoloading mechanism of PHP classes following PSR-4.

Router
Phalcon\Mvc\Router provides advanced routing capabilities.

MVC

Build single and multi-module applications with ease and pleasure. Using the file structure, scheme and patterns you already know.

```
single/
    app/
        controllers/
        models/
        views/
    public/
        css/
        img/
        js/
```

```
multiple/
    apps/
      frontend/
         controllers/
         models/
         views/
         Module.php
      backend/
         controllers/
         models/
         views/
         Module.php
      public/
      ../
```

BASIC FEATURES

Low overhead
Low memory consumption and CPU compared to traditional frameworks

MVC & HMVC
Modules, components, models, views and controllers

Dependency Injection
Dependency Injection and Location of services and it's itself a container for them.

Rest
In this case, you can use either a micro or full stack application to meet your goal. In addition, a powerful set of HTTP helpers.

Autoloader
Provides the autoloading mechanism of PHP classes following PSR-4.

Router
Phalcon\Mvc\Router provides advanced routing capabilities.

Dependency Injection

Phalcon is built upon a powerful yet easy to understand and use pattern called Dependency Injection. Initialize or define services once - and use them virtually anywhere throughout the application.

```
// Create the Dependency Injector Container
$di = new Phalcon\DI();

//Register classes, functions, components
$di->set("request", new Phalcon\Http\Request());

..

// Use anywhere else in code
$request = $di->getShared('request');
```

BASIC FEATURES

Low overhead
Low memory consumption and CPU compared to traditional frameworks

MVC & HMVC
Modules, components, models, views and controllers

Dependency Injection
Dependency Injection and Location of services and it's itself a container for them.

Rest
In this case, you can use either a micro or full stack application to meet your goal. In addition, a powerful set of HTTP helpers.

Autoloader
Provides the autoloading mechanism of PHP classes following PSR-4.

Router
Phalcon\Mvc\Router provides advanced routing capabilities.

Restful Services

Writing REST servers and applications has never been easier. No boilerplate. Simple services will fit in one file.

```php
use Phalcon\Mvc\Micro;

$app = new Micro();

// Returning data in JSON
$app->get(
    '/check/status',
    function () {
        return $this->response->setJsonContent(
            [
                'status' => 'important',
            ]
        );
    }
);

$app->handle();
```

BASIC FEATURES

Low overhead
Low memory consumption and CPU compared to traditional frameworks

MVC & HMVC
Modules, components, models, views and controllers

Dependency Injection
Dependency Injection and Location of services and it's itself a container for them.

Rest
In this case, you can use either a micro or full stack application to meet your goal. In addition, a powerful set of HTTP helpers.

Autoloader
Provides the autoloading mechanism of PHP classes following PSR-4.

Router
Phalcon\Mvc\Router provides advanced routing capabilities.

Autoloader

Register namespaces, prefixes, directories or classes. Take advantage of the autoloader events and maintain full control over what files are loaded and from where.

```php
use Phalcon\Loader;

// Creates the autoloader
$loader = new Loader();

// Register some namespaces
$loader->registerNamespaces(
    [
        'Example\Base'    => 'vendor/example/base/',
        'Example\Adapter' => 'vendor/example/adapter/',
        'Example'         => 'vendor/example/',
    ]
);

// Register autoloader
$loader->register();
```

BASIC FEATURES

Low overhead
Low memory consumption and CPU compared to traditional frameworks

MVC & HMVC
Modules, components, models, views and controllers

Dependency Injection
Dependency Injection and Location of services and it's itself a container for them.

Rest
In this case, you can use either a micro or full stack application to meet your goal. In addition, a powerful set of HTTP helpers.

Autoloader
Provides the autoloading mechanism of PHP classes following PSR-4.

Router
Phalcon\Mvc\Router provides advanced routing capabilities.

Router

Routing as it supposed to be. Nothing more. Nothing less.

```php
// Create the router
$router = new \Phalcon\Mvc\Router();

// Define a route
$router->add(
   '/admin/users/my-profile',
   [
       'controller' => 'users',
       'action'     => 'profile',
   ]
);
```

DATA & STORAGE

- **ORM**
  Object Relational Mapping
- **PHQL**
  The powerful and secure Phalcon Query Language, PHQL
- **ODM for MongoDB**
  Object Document Mapping for MongoDB
- **Transactions**
  Transactions in Phalcon allows to keep the data integrity safe.
- **Cache**
  Improve your performance with many of the backend caches that Phalcon provides

ORM

A powerful ORM is provided by Phalcon allowing you to manipulate database records as classes and objects. MySQL, PostgreSQL and SQLite are supported out of the box.

```php
use Phalcon\Mvc\Model;

class Robots extends Model
{
    public $id;

    public $name;

    public function initialize()
    {
        $this->hasMany('id', 'RobotsParts', 'robots_id');
    }
}
```

DATA & STORAGE

- **ORM**
  Object Relational Mapping
- **PHQL**
  The powerful and secure Phalcon Query Language, PHQL
- **ODM for MongoDB**
  Object Document Mapping for MongoDB
- **Transactions**
  Transactions in Phalcon allows to keep the data integrity safe.
- **Cache**
  Improve your performance with many of the backend caches that Phalcon provides

PHQL

PHQL is a high-level, object-oriented SQL dialect that allows to write queries using a standardized SQL-like language. PHQL is implemented as a parser (written in C) that translates syntax in that of the target RDBMS. To achieve the highest performance possible, Phalcon provides a parser that uses the same technology as SQLite. This technology provides a small in-memory parser with a very low memory footprint that is also thread-safe.

```php
$phql  = 'SELECT * FROM Formula\Cars ORDER BY Formula\Cars.name';
$query = $manager->createQuery($phql);

$phql  = 'SELECT Formula\Cars.name FROM Formula\Cars ORDER BY Formula\Cars.name';
$query = $manager->createQuery($phql);

$phql  = 'SELECT c.name FROM Formula\Cars c ORDER BY c.name';
$query = $manager->createQuery($phql);

$phql = 'SELECT c.* FROM Cars AS c ORDER BY c.name';
$cars = $manager->executeQuery($phql);
foreach ($cars as $car) {
    echo "Name: ", $car->name, "\n";
}
```

DATA & STORAGE

- **ORM**
  Object Relational Mapping
- **PHQL**
  The powerful and secure Phalcon Query Language, PHQL
- **ODM for MongoDB**
  Object Document Mapping for MongoDB
- **Transactions**
  Transactions in Phalcon allows to keep the data integrity safe.
- **Cache**
  Improve your performance with many of the backend caches that Phalcon provides

ODM for MongoDB

In addition to its ability to map tables in relational databases, Phalcon can map documents to a MongoDB database. The ODM offers a CRUD functionality, events, validations among other services.

```
// How many robots are there?
$robots = Robots::find();
echo "There are count($robots) robots\n";

// How many mechanical robots are there?$robots = Robots::find(
    array(
        array(
            "type" => "mechanical"
        )
    )
);
echo "There are count($robots) robots\n";
```

DATA & STORAGE

- **ORM**
  Object Relational Mapping
- **PHQL**
  The powerful and secure Phalcon Query Language, PHQL
- **ODM for MongoDB**
  Object Document Mapping for MongoDB
- **Transactions**
  Transactions in Phalcon allows to keep the data integrity safe.
- **Cache**
  Improve your performance with many of the backend caches that Phalcon provides

Transactions

When a process performs multiple database operations, it might be important that each step is completed successfully so that data integrity can be maintained. Transactions offer the ability to ensure that all database operations have been executed successfully before the data is committed to the database.

```php
use Phalcon\Mvc\Model\Transaction\Failed as TxFailed;
use Phalcon\Mvc\Model\Transaction\Manager as TxManager;

try {

    // Create a transaction manager
    $manager     = new TxManager();

    // Request a transaction
    $transaction = $manager->get();

    // Get the robots to be deleted
    foreach (Robots::find("type = 'mechanical'") as $robot) {
        $robot->setTransaction($transaction);
        if ($robot->delete() == false) {
            // Something's gone wrong, we should rollback the transaction
            foreach ($robot->getMessages() as $message) {
                $transaction->rollback($message->getMessage());
            }
        }
    }

    // Everything's gone fine, let's commit the transaction
    $transaction->commit();

    echo "Robots were deleted successfully!";

} catch (TxFailed $e) {
    echo "Failed, reason:  ", $e->getMessage();
}
```

DATA & STORAGE

- **ORM**
  Object Relational Mapping
- **PHQL**
  The powerful and secure Phalcon Query Language, PHQL
- **ODM for MongoDB**
  Object Document Mapping for MongoDB
- **Transactions**
  Transactions in Phalcon allows to keep the data integrity safe.
- **Cache**
  Improve your performance with many of the backend caches that Phalcon provides

Cache

The cache component allows faster access to frequently used or already processed data. It supports many backends such as Redis, Memcached, Mongo, Files, Apc and more

```php
use Phalcon\Cache\Frontend\Data as FrontendData;
use Phalcon\Cache\Backend\Memcache as BackendMemcache;

// Set the models cache service
$di->set('modelsCache', function () {

    // Cache data for one day by default
    $frontCache = new FrontendData(
        [
            'lifetime' => 86400,
        ]
    );

    // Memcached connection settings
    $cache = new BackendMemcache(
        $frontCache,
        [
            'host' => 'localhost',
            'port' => '11211',
        ]
    );

    return $cache;
});
```

VIEWS & FRONTEND

Template Engines
Views represent the user interface of your application

Template Engine (Volt)
A template engine inspired by Jinja but built in C for PHP

i18n
Translate your applications to many languages easily

Forms Builder
Easily create HTML forms

Flash messages
Flash messages are used to notify the user about the state of actions.

Template Engines

Views represent the user interface of your application. Views are often HTML files with embedded PHP code that perform tasks related solely to the presentation of the data. Views handle the job of providing data to the web browser or other tool that is used to make requests from your application.

```
<html>
    <body>
    <div class='top'><?php $this->partial('shared/ad_banner'); ?></div>
    <div class='content'>
        <h1>Robots</h1>
        <p>Check out our specials for robots:</p>
        ...
    </div>
    <div class='footer'><?php $this->partial('shared/footer'); ?></div>
    </body>
</html>
```

VIEWS & FRONTEND

Template Engines
Views represent the user interface of your application

Template Engine (Volt)
A template engine inspired by Jinja but built in C for PHP

i18n
Translate your applications to many languages easily

Forms Builder
Easily create HTML forms

Flash messages
Flash messages are used to notify the user about the state of actions.

Template Engine (Volt)

Volt is an ultra-fast and designer friendly templating language written in Zephir/C for PHP. It provides you a set of helpers to write views in an easy way. Volt is highly integrated with other components of Phalcon, just as you can use it as a stand-alone component in your applications.

```
{# app/views/products/show.volt #}
{% block last_products %}
{% for product in products %}
    * Name: {{ product.name|e }}
    {% if product.status == 'Active' %}
       Price: {{ product.price + product.taxes/100}}
    {% endif  %}
{% endfor  %}
{% endblock %}
```

VIEWS & FRONTEND

- **Template Engines**
  Views represent the user interface of your application
- **Template Engine (Volt)**
  A template engine inspired by Jinja but built in C for PHP
- **i18n**
  Translate your applications to many languages easily
- **Forms Builder**
  Easily create HTML forms
- **Flash messages**
  Flash messages are used to notify the user about the state of actions.

i18n

The component Phalcon\Translate aids in creating multilingual applications. Applications using this component, display content in different languages, based on the user's chosen language supported by the application.

```
// app/messages/en.php
$messages = [
    'hi'      => 'Hello',
    'bye'     => 'Good Bye',
    'hi-name' => 'Hello %name%',
    'song'    => 'This song is %song%'
];

// app/messages/es.php
$messages = [
    'hi'      => 'Hola',
    'bye'     => 'Adiós',
    'hi-name' => 'Hola %name%',
    'song'    => 'Esta canción es %song%'
];

use Phalcon\Mvc\Controller;
use Phalcon\Translate\Adapter\NativeArray;

// UserController.php
class UserController extends Controller
{
    protected function getTranslation()
    {
        // Ask browser what is the best language
        $language = $this->request->getBestLanguage();

        // Check if we have a translation file for that lang
        if (file_exists('app/messages/' . $language . '.php')) {
            require 'app/messages/' . $language . '.php';
        } else {
            // Fallback to some default           require 'app/messages/en.ph
p';
        }

        // Return a translation object
        return new NativeArray(
            array(
                'content' => $messages
            )
        );
    }

    public function indexAction()
    {
        $this->view->name = 'Mike';
        $this->view->t    = $this->getTranslation();
    }
}
// user.volt
<p><?php echo $t->_('hi'), ' ', $name; ?></p>
```

VIEWS & FRONTEND

- **Template Engines**
  Views represent the user interface of your application
- **Template Engine (Volt)**
  A template engine inspired by Jinja but built in C for PHP
- **i18n**
  Translate your applications to many languages easily
- **Forms Builder**
  Easily create HTML forms
- **Flash messages**
  Flash messages are used to notify the user about the state of actions.

Forms Builder

Each element in the form can be rendered as required by the developer. Internally, Phalcon\Tag is used to produce the correct HTML for each element and you can pass additional HTML attributes as the second parameter of render():

```php
use Phalcon\Forms\Form;
use Phalcon\Forms\Element\Text;
use Phalcon\Forms\Element\Select;

$form = new Form();

$form->add(new Text('name'));

$form->add(new Text('telephone'));

$form->add(
    new Select(
        'telephoneType',
        array(
            'H' => 'Home',
            'C' => 'Cellphone'
        )
    )
);
```

VIEWS & FRONTEND

Template Engines
Views represent the user interface of your application

Template Engine (Volt)
A template engine inspired by Jinja but built in C for PHP

i18n
Translate your applications to many languages easily

Forms Builder
Easily create HTML forms

Flash messages
Flash messages are used to notify the user about the state of actions.

Flash messages

Use the flash notifier to show notifications to a user in a web application:

```
use Phalcon\Mvc\Controller;

class PostsController extends Controller
{
    public function saveAction()
    {
        $this->flash->error('there were errors in this form');
        $this->flash->success('yes!, everything went very smoothly');
        $this->flash->notice('this is a notice for users');
        $this->flash->warning('this is just a warning');
    }
}
```

MORE...

ACL
Access Control List allows users to access the modules they're authorized to

Sharding
Connect, store and retrieve data from many database systems at the same time

Crypt
Encrypt/Decrypt important data to keep them safe from unauthorized third-parties

Events
Extend the most of the framework components by setting 'hook points'. Create your own events and make your application more flexible and powerful

Queueing - background process
Use the built-in queueing system in Phalcon to schedule jobs and reduce the load of your web server

ACL

This is how you can built the access control list (ACL):

```php
use Phalcon\Acl;
use Phalcon\Acl\Role;
use Phalcon\Acl\Adapter\Memory as AclList;

// Create the ACL
$acl = new AclList();

// The default action is DENY access
$acl->setDefaultAction(Acl::DENY);

// Register two roles, Users is registered users
// and guests are users without a defined identity
$roles = array(
    'users'  => new Role('Users'),
    'guests' => new Role('Guests')
);

foreach ($roles as $role) {
    $acl->addRole($role);
}
```

MORE...

ACL
Access Control List allows users to access the modules they're authorized to

Sharding
Connect, store and retrieve data from many database systems at the same time

Crypt
Encrypt/Decrypt important data to keep them safe from unauthorized third-parties

Events
Extend the most of the framework components by setting 'hook points'. Create your own events and make your application more flexible and powerful

Queueing - background process
Use the built-in queueing system in Phalcon to schedule jobs and reduce the load of your web server

Sharding

Attach models to different databases

```php
use Phalcon\Db\Adapter\Pdo\Mysql as MysqlPdo;
use Phalcon\Db\Adapter\Pdo\PostgreSQL as PostgreSQLPdo;

// This service returns a MySQL database
$di->set(
    'dbMysql',
    function () {
        return new MysqlPdo(
            [
                'host'     => 'localhost',
                'username' => 'root',
                'password' => 'secret',
                'dbname'   => 'invo',
            ]
        );
    }
);

// This service returns a PostgreSQL database
$di->set(
    'dbPostgres',
    function () {
        return new PostgreSQLPdo(
            [
                'host'     => 'localhost',
                'username' => 'postgres',
                'password' => '',
                'dbname'   => 'invo',
            ]
        );
    }
);
```

MORE...

ACL
Access Control List allows users to access the modules they're authorized to

Sharding
Connect, store and retrieve data from many database systems at the same time

Crypt
Encrypt/Decrypt important data to keep them safe from unauthorized third-parties

Events
Extend the most of the framework components by setting 'hook points'. Create your own events and make your application more flexible and powerful

Queueing - background process
Use the built-in queueing system in Phalcon to schedule jobs and reduce the load of your web server

Encryption

Phalcon provides encryption facilities via the Phalcon\Crypt component. This class offers simple object-oriented wrappers to the openssl PHP's encryption library.

```php
use Phalcon\Crypt;

// Create an instance
$crypt     = new Crypt();

$key       = 'This is a secret key (32 bytes).';
$text      = 'This is the text that you want to encrypt.';

$encrypted = $crypt->encrypt($text, $key);

echo $crypt->decrypt($encrypted, $key);
```

MORE...

ACL
Access Control List allows users to access the modules they're authorized to

Sharding
Connect, store and retrieve data from many database systems at the same time

Crypt
Encrypt/Decrypt important data to keep them safe from unauthorized third-parties

Events
Extend the most of the framework components by setting 'hook points'. Create your own events and make your application more flexible and powerful

Queueing - background process
Use the built-in queueing system in Phalcon to schedule jobs and reduce the load of your web server

Events Management

An EventsManager allows us to attach listeners to a particular type of event. The type that interests us now is 'dispatch'. The following code filters all events produced by the Dispatcher:

```php
use Phalcon\Mvc\Dispatcher;
use Phalcon\Events\Manager as EventsManager;

$di->set('dispatcher', function () {

    // Create an events manager
    $eventsManager = new EventsManager();

    // Listen for events produced in the dispatcher using the Security plugin
    $eventsManager->attach('dispatch:beforeExecuteRoute', new SecurityPlugin);

    // Handle exceptions and not-found exceptions using NotFoundPlugin
    $eventsManager->attach('dispatch:beforeException', new NotFoundPlugin);

    $dispatcher = new Dispatcher();

    // Assign the events manager to the dispatcher
    $dispatcher->setEventsManager($eventsManager);

    return $dispatcher;
});
```

MORE...

ACL
Access Control List allows users to access the modules they're authorized to

Sharding
Connect, store and retrieve data from many database systems at the same time

Crypt
Encrypt/Decrypt important data to keep them safe from unauthorized third-parties

Events
Extend the most of the framework components by setting 'hook points'. Create your own events and make your application more flexible and powerful

Queueing - background process
Use the built-in queueing system in Phalcon to schedule jobs and reduce the load of your web server

Queueing - background process

Activities like processing videos, resizing images or sending emails aren't suitable to be executed online or in real time because it may slow the loading time of pages and severely impact the user experience. The best solution here is to implement background jobs. The web application puts jobs into a queue and which will be processed separately.

```php
use Phalcon\Queue\Beanstalk;

// Connect to the queue
$queue = new Beanstalk(
    [
        'host' => '192.168.0.21',
        'port' => '11300',
    ]
);

// Insert the job in the queue
$queue->put(
    [
        'processVideo' => 4871,
    ]
);
```