PHP
File System, CACHE e NAMESPACE

Prof. Ing. Loris Penserini

elpense@gmail.com

Cache and website performance

Il meccanismo di CACHE è molto usato per aumentare le prestazioni di un sito web.

Ci sono svariate situazioni in cui si accede a servizi web che richiamano metodi che accedono molto frequentemente alle risorse/dati. Tuttavia, a fronte di queste richieste di dati, la risposta al client è sempre la stessa.

Per esempio:

- Richiesta di dettagli sempre sullo stesso prodotto dello shop online che prevede l'interrogazione di un database
- Lettura dello stesso file XML per produrre la risposta ad un client
- Ecc.

Alternative PHP Cache (APC)

Aumentare le prestazioni di un sito Web PHP.

APC è un ottimo sistema di caching di op-code per PHP, e quindi aiuta a velocizzare un sito web. APC Cache aiuta a bypassare i passaggi di analisi e compilazione e riduce al minimo la richiesta web al server.

APC non è più compatibile con la versione php 5.5+, quindi si utilizza una versione aggiornata: **APCu**.

Installare il modulo APCu

Il modulo/libreria lo potete trovare qui:

- https://pecl.php.net/package/apcu

E' un progetto open-source, e su GitHub trovate tutti i sorgenti.

Per utilizzarlo con XAMPP, cioè in WINDOWS:

- Copiare la «**php_apcu.dll**» in **c:\xampp\php\ext**
- Aggiungere in «**php.ini**» la riga: **extension=php_apcu.dll**

Alcuni dettagli di configurazione, benché inutili in molti casi, possono essere letti qui:

- https://www.php.net/manual/en/apcu.configuration.php

Verificare la presenza di APCu

Una volta eseguite le operazioni precedenti, possiamo testare se il Web Server è stato correttamente configurato per l'utilizzo del modulo APCu. Per cui basta eseguire il seguente script PHP:

```
<?php
    // Ottiene informazioni di sistema per l'ambiente PHP
    phpinfo();
?>
```

L'output consiste in una tabella con i dettagli delle applicazioni/servizi installati per il funzionamento del Web Server e del PHP…

Ambiente PHP

apcu

APCu Support	Enabled
Version	5.1.21
APCu Debugging	Disabled
MMAP Support	Disabled
Serialization Support	php
Build Date	Oct 7 2021 11:39:14

Directive	Local Value	Master Value
apc.coredump_unmap	Off	Off
apc.enable_cli	Off	Off
apc.enabled	On	On
apc.entries_hint	4096	4096
apc.gc_ttl	3600	3600
apc.preload_path	*no value*	*no value*
apc.serializer	php	php
apc.shm_segments	1	1
apc.shm_size	32M	32M
apc.slam_defense	Off	Off
apc.smart	0	0
apc.ttl	0	0
apc.use_request_time	Off	Off

bcmath

BCMath support	enabled

Esempio di utilizzo di APCu

Consideriamo di dover accedere ai dei dati del servizio Web molto frequentemente: cioè una risposta XML da inviare ai client che nella maggior parte dei casi è sempre la stessa.

Per cui, faremo in modo che tale risposta sia memorizzata nella cache fino a quando non cambia il file XML; per cui, nella maggior parte dei casi, la risposta XML verrà prelevata dalla cache senza appesantire il web server con l'accesso alla risorsa del file system.

Prima di mostrare l'esempio, occorre passare in rassegna alcune istruzioni che verranno utilizzate…

Alcuni dettagli di APCu

- APCu Functions
  - apcu_add — Cache a new variable in the data store
  - apcu_cache_info — Retrieves cached information from APCu's data store
  - apcu_cas — Updates an old value with a new value
  - apcu_clear_cache — Clears the APCu cache
  - apcu_dec — Decrease a stored number
  - apcu_delete — Removes a stored variable from the cache
  - apcu_enabled — Whether APCu is usable in the current environment
  - apcu_entry — Atomically fetch or generate a cache entry
  - apcu_exists — Checks if entry exists
  - apcu_fetch — Fetch a stored variable from the cache
  - apcu_inc — Increase a stored number
  - apcu_key_info — Get detailed information about the cache key
  - apcu_sma_info — Retrieves APCu Shared Memory Allocation information
  - apcu_store — Cache a variable in the data store

- APCUIterator — The APCUIterator class
  - APCUIterator::__construct — Constructs an APCUIterator iterator object
  - APCUIterator::current — Get current item
  - APCUIterator::getTotalCount — Get total count
  - APCUIterator::getTotalHits — Get total cache hits
  - APCUIterator::getTotalSize — Get total cache size
  - APCUIterator::key — Get iterator key
  - APCUIterator::next — Move pointer to next item
  - APCUIterator::rewind — Rewinds iterator
  - APCUIterator::valid — Checks if current position is valid

Per ulteriori dettagli:
https://www.php.net/apcu

Alcune Operazioni su File System

Funzione	Significato
time()	Restituisce il timestamp in secondi
filemtime("file.txt")	Restituisce il timestamp dell'ultima modifica
fileatime("file.txt")	Restituisce il timestamp dell'ultimo accesso
fopen($filename, 'w+')	$filename -> standard URL Mode: w -> file riscritto se esistente r -> letto dall'inizio w+ -> lettura/scrittura e sovrascrive a+ -> lettura/scrittura dalla fine …
fwrite($filename, $file_content)	Scrive nella risorsa puntata da $filename il testo contenuto in $file_content
file_get_contents($filename)	Restituisce il contenuto del file
fclose($filename)	Chiude il canale con la risorsa

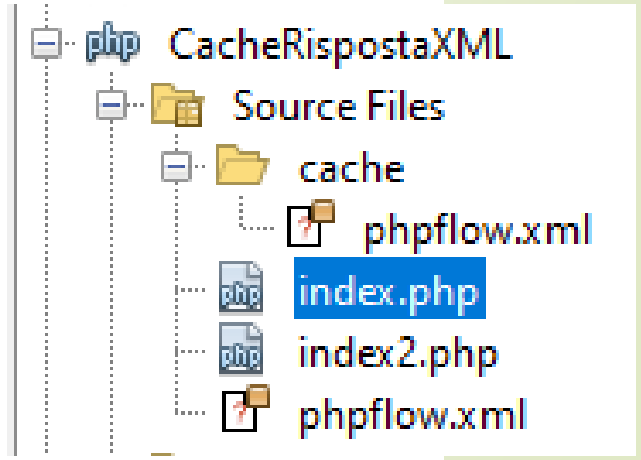
Esempio di utilizzo di CACHE di file

L'esempio che segue è un semplice esempio di utilizzo di cache di file, utilizzando una cartella del filesystem.

```php
<?php

 //* Funzione con file
function checkCache() {
    $path_cache = 'cache/phpflow.xml';
    $path_newFile = 'phpflow.xml';
    //if ((!file_exists($path) || time() - filemtime($path) > 30) && $cache = fopen($path, 'w+')) {
    if (!file_exists($path_cache) || time() - filemtime($path_newFile) < 20){
        $cache = fopen($path_cache, 'w+');
        fwrite($cache, file_get_contents($path_newFile));
        echo "Copia del NUOVO File in cache<br>";
        fclose($cache);
        return file_get_contents($path_cache);
    } else {
        $cache = fopen($path_cache, 'r');
        fclose($cache);
        echo "<br>Si legge il file in cache<br>";
        return file_get_contents($path_cache);
    }
}
$checkCache = checkCache();
echo $checkCache."<br>";

```



Project Work 13

Una volta eseguito il codice, provate a svolgere queste operazioni:

- Editate il file «phpflow.xml» e, subito dopo aver salvato, aggiornate la pagina del browser → verrà chiesto al Server Web di rieseguire lo script…
- Cancellate il file «phpflow.xml» che si trova dento la cartella «cache» e, subito dopo aggiornate la pagina del browser →verrà richiesto al Server Web di ricopiare il file nella cartella «cache»

Scenario di utilizzo di APCu

L'esempio che segue è un semplice esempio di utilizzo di APCu, ma unisce anche qualche istruzione di manipolazione del file system.

Il progetto consiste nel caricare un file XML nella cache (gestita da APCu), e aggiornare la cache solo se il file viene modificato. Nel caso in cui il contenuto del file rimane invariato, il Server Web continua a restituire al client il contenuto della cache, cioè senza accedere al file system.

Questo meccanismo, nella realtà dei server web accelera di molto le prestazioni.

Esempio semplice con APCu

L'esempio che segue è un semplice esempio di utilizzo di cache di file, utilizzando APCu

```php
<?php
 // Funzione con APCu
function checkCache() {
    $oriFile = 'phpflow.xml';
    $cache = file_get_contents($oriFile);

    if ((!apcu_exists('phpflow')) || apcu_fetch('phpflow') != $cache) {
        echo 'Inserisce il file XML modificato nella cache ';
        echo '<br>';

        $cache = file_get_contents($oriFile);
        apcu_store('phpflow', $cache, 0);
        return $cache;

    } else {
        echo '** continua ad usare il file XML nella cache **';
        echo '<br>';
        print(apcu_fetch('phpflow'));
    }
}
$checkCache = checkCache();
echo $checkCache."!!";
?>
```

Project Work 13

Provate a dare un valore al parametro TTL, per es. 10sec.

Poi aggiornate il browser, ogni cosa accade?

Project Work 14

Considerate l'esempio di utilizzo di APCu precedentemente illustrato.

Fornirne un'interpretazione ad oggetti.

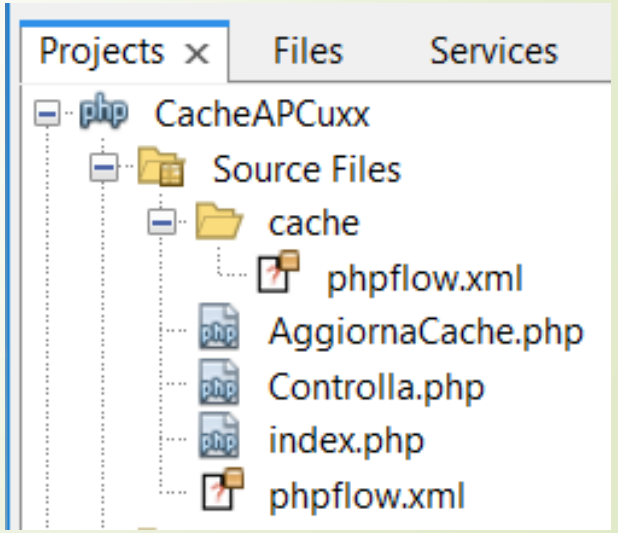
OOP con l'utilizzo di APCu: «index.php»

```php
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        include 'AggiornaCache.php';
        include 'Controlla.php';

        $fileToCache = 'cache/phpflow.xml';
        $checkForNewFile = 'phpflow.xml';

        $controllaFile = new Controlla($checkForNewFile,$fileToCache);
        echo $controllaFile->checkFile();
        ?>
    </body>
</html>
```

NetBeans IDE project structure



OOP con l'utilizzo di APCu: «Controlla.php»

```php
class Controlla extends AggiornaCache{
    private $fileToCache = 'cache/phpflow.xml';
    private $checkForNewFile = 'phpflow.xml';
    //private bool $modificato = false;

    function __construct($checkForNewFile, $fileToCache){
        $this->checkForNewFile = $checkForNewFile;
        $this->fileToCache = $fileToCache;
        parent::storeFile($fileToCache);
    }

    function checkFile() {
        //"< 10" si mette solamente per dare la possibilità di far vedere
        //didatticamente l'effetto di modifica del file...
        if (!file_exists($this->fileToCache) || time() - filemtime($this->checkForNewFile) < 10){
            //***IL FILE E' STATO MODIFICATO***
            $cache = fopen($this->fileToCache, 'w+');
            fwrite($cache, file_get_contents($this->checkForNewFile));
            fclose($cache);
            $rispostaCache = parent::updateCache($this->checkForNewFile);
            return $rispostaCache;
        } else {
            //FILE INVARIATO, QUINDI SI CONTINUA A LEGGERE DALLA CACHE
            return "Nessuna modifica al file. Quindi, CACHE not updated!<br> -- Leggo cache: ".parent::readCache();
        }
    }
}
```

OOP con l'utilizzo di APCu: «AggiornaCache.php»

```php
class AggiornaCache{
    private $fileToCache = "";
    private $cache = "";

    function __construct($fileToCache) {
        $this->fileToCache = $fileToCache;
        $this->cache = file_get_contents($fileToCache);
        apcu_store($fileToCache, $this->cache, 0);//Aggiorna la CACHE, in modo permanente
    }

    function storeFile($fileToCache) {
        $this->fileToCache = $fileToCache;
        $this->cache = file_get_contents($fileToCache);
        apcu_store($fileToCache, $this->cache, 0);//Aggiorna la CACHE, in modo permanente
    }

    function readCache(){
        return apcu_fetch($this->fileToCache);
    }

    function updateCache($newFileToCache) {
        $newCache = file_get_contents($newFileToCache);

        if ((!apcu_exists($newFileToCache)) || apcu_fetch($this->fileToCache) != $newCache) {
            apcu_store($this->fileToCache, $newCache, 0);
            return "Il File è stato modificato! Cache updated!";
        } else {
            return "Cache not updated!";
        }
    }
}
```

Project Work 13

Realizzare un benchmark per testare quanto sia più veloce tenere file/dati in cache, piuttosto che leggerli dal filesystem oppure leggerli da database.

Namespace

Quando capita di aver bisogno di più librerie che hanno nomi di classi uguali, tecnicamente si dice che le librerie vengono usate nello stesso spazio dei nomi, per cui classi con stesso nome causano problemi.

I namespace risolvono questo problema.

In PHP, gli spazi dei nomi sposano lo stesso meccanismo delle directory del file system del sistema operativo: per cui due file con lo stesso nome possono coesistere se in directory separate. Allo stesso modo, due classi PHP con lo stesso nome possono coesistere in spazi dei nomi PHP separati.

GRAZIE!