



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	<i>no value</i>	<i>no value</i>
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 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>

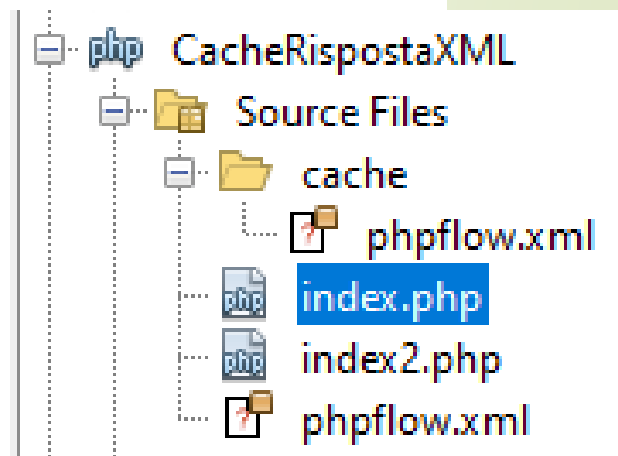
Alcune Operazioni su File System

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.

```
12 <?php
13
14 /* Funzione con file
15 function checkCache() {
16     $path_cache = 'cache/phpflow.xml';
17     $path_newFile = 'phpflow.xml';
18     //if ((!file_exists($path) || time() - filemtime($path) > 30) && $cache = fopen($path, 'w+')) {
19     if (!file_exists($path_cache) || time() - filemtime($path_newFile) < 20){
20         $cache = fopen($path_cache, 'w+');
21         fwrite($cache, file_get_contents($path_newFile));
22         echo "Copia del NUOVO File in cache<br>";
23         fclose($cache);
24         return file_get_contents($path_cache);
25     } else {
26         $cache = fopen($path_cache, 'r');
27         fclose($cache);
28         echo "<br>Si legge il file in cache<br>";
29         return file_get_contents($path_cache);
30     }
31 }
32 $checkCache = checkCache();
33 echo $checkCache."<br>";
34
```



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 dentro 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

```
12 <?php
13 // Funzione con APCu
14 function checkCache() {
15     $oriFile = 'phpflow.xml';
16     $cache = file_get_contents($oriFile);
17
18     if ((!apcu_exists('phpflow')) || apcu_fetch('phpflow') != $cache) {
19         echo 'Inserisce il file XML modificato nella cache ';
20         echo '<br>';
21
22         $cache = file_get_contents($oriFile);
23         apcu_store('phpflow', $cache, 0);
24         return $cache;
25
26     } else {
27         echo '** continua ad usare il file XML nella cache **';
28         echo '<br>';
29         print(apcu_fetch('phpflow'));
30     }
31 }
32 $checkCache = checkCache();
33 echo $checkCache."!!";
34 ?>
```

Project Work 13

**Provate a dare un valore al parametro TTL, per es. 10sec.
Poi aggiornate il browser, ogni cosa accade?**

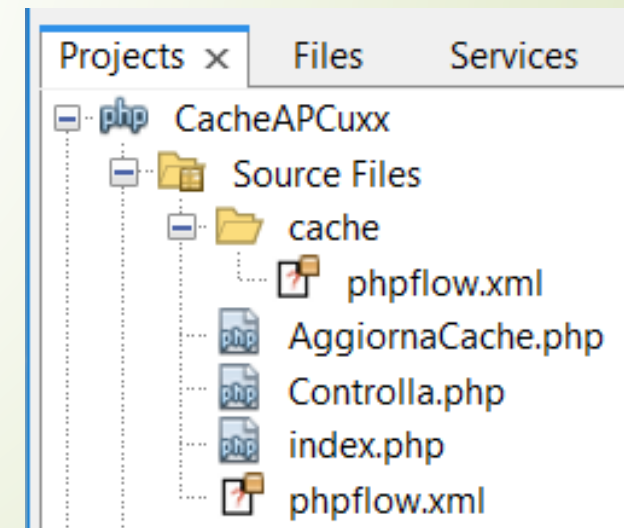
Project Work 14

**Considerate l'esempio di utilizzo di APCu precedentemente illustrato.
Fornirne un'interpretazione ad oggetti.**

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

```
6 <html>
7 <head>
8     <meta charset="UTF-8">
9     <title></title>
10 </head>
11 <body>
12     <?php
13     include 'AggiornaCache.php';
14     include 'Controlla.php';
15
16     $fileToCache = 'cache/phpflow.xml';
17     $checkForNewFile = 'phpflow.xml';
18
19     $controllaFile = new Controlla($checkForNewFile,$fileToCache);
20     echo $controllaFile->checkFile();
21     ?>
22 </body>
23 </html>
```

NetBeans IDE project structure



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

```
13 class Controlla extends AggiornaCache{
14     private $fileToCache = 'cache/phpflow.xml';
15     private $checkForNewFile = 'phpflow.xml';
16     //private bool $modificato = false;
17
18     function __construct($checkForNewFile, $fileToCache){
19         $this->checkForNewFile = $checkForNewFile;
20         $this->fileToCache = $fileToCache;
21         parent::storeFile($fileToCache);
22     }
23
24     function checkFile() {
25         //"< 10" si mette solamente per dare la possibilità di far vedere
26         //didatticamente l'effetto di modifica del file...
27         if (!file_exists($this->fileToCache) || time() - filemtime($this->checkForNewFile) < 10){
28             /**IL FILE E' STATO MODIFICATO**/
29             $cache = fopen($this->fileToCache, 'w+');
30             fwrite($cache, file_get_contents($this->checkForNewFile));
31             fclose($cache);
32             $rispostaCache = parent::updateCache($this->checkForNewFile);
33             return $rispostaCache;
34         } else {
35             //FILE INVARIATO, QUINDI SI CONTINUA A LEGGERE DALLA CACHE
36             return "Nessuna modifica al file. Quindi, CACHE not updated!<br> -- Leggo cache: ".parent::readCache();
37         }
38     }
39 }
```

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

```
14 class AggiornaCache{
15     private $fileToCache = "";
16     private $cache = "";
17
18     function __construct($fileToCache) {
19         $this->fileToCache = $fileToCache;
20         $this->cache = file_get_contents($fileToCache);
21         apcu_store($fileToCache, $this->cache, 0); //Aggiorna la CACHE, in modo permanente
22     }
23
24     function storeFile($fileToCache) {
25         $this->fileToCache = $fileToCache;
26         $this->cache = file_get_contents($fileToCache);
27         apcu_store($fileToCache, $this->cache, 0); //Aggiorna la CACHE, in modo permanente
28     }
29
30     function readCache() {
31         return apcu_fetch($this->fileToCache);
32     }
33
34     function updateCache($newFileToCache) {
35         $newCache = file_get_contents($newFileToCache);
36
37         if ((!apcu_exists($newFileToCache)) || apcu_fetch($this->fileToCache) != $newCache) {
38             apcu_store($this->fileToCache, $newCache, 0);
39             return "Il File è stato modificato! Cache updated!";
40         } else {
41             return "Cache not updated!";
42         }
43     }
44 }
```



Project Work 13

Nell'esempio appena visto, con l'utilizzo di APCu ad oggetti, pensare ad una soluzione alternativa che faccia un uso efficace dei TRAIT.

PW-13: soluzione proposta

Nell'esempio appena visto, con l'utilizzo di APCu ad oggetti, pensare ad una soluzione alternativa che faccia un uso efficace dei TRAIT.

SOLUZ.

Nell'esempio appena visto, si modifica solo la classe «**AggiornaCache**» esternalizzando, dentro due trait, i due metodi «**storeFile(\$fileToCache)**» e «**readCache()**» come segue...

PW-13: i due trait

I trait riportano i metodi corrispondenti della classe «AggiornaCache».

```
12 [-] trait traitReadCache {  
13     [-] public function readCache() {  
15         return apcu_fetch($this->fileToCache);  
16     }  
17 }
```

```
12 [-] trait storeFile {  
13     [-] function storeFile($fileToCache) {  
14         $this->fileToCache = $fileToCache;  
15         $this->cache = file_get_contents($fileToCache);  
16         apcu_store($fileToCache, $this->cache, 0); //Aggiorna la CACHE, in modo permanente  
17     }  
18 }
```

PW-13: «AggiornaCache»

```
13 include 'traitReadCache.php';
14 include 'storeFile.php';
15
16 class AggiornaCache{
17     use traitReadCache,storeFile;
18     private $fileToCache = "";
19     private $cache = "";
20
21     function __construct($fileToCache) {
22         $this->fileToCache = $fileToCache;
23         $this->cache = file_get_contents($fileToCache);
24         apcu_store($fileToCache, $this->cache, 0);//Aggiorna la CACHE, in modo permanente
25     }
26
27     /*
28     function storeFile($fileToCache) {
29         $this->fileToCache = $fileToCache;
30         $this->cache = file_get_contents($fileToCache);
31         apcu_store($fileToCache, $this->cache, 0);//Aggiorna la CACHE, in modo permanente
32     }
33
34     function readCache(){
35         return apcu_fetch($this->fileToCache);
36     }
37     */
38
39     function updateCache($newFileToCache) {
40         $newCache = file_get_contents($newFileToCache);
41
42         if (!apcu_exists($newFileToCache) || apcu_fetch($this->fileToCache) != $newCache) {
43             apcu_store($this->fileToCache, $newCache, 0);
44             return "Il File è stato modificato! Cache updated!";
45         } else {
46             return "Cache not updated!";
47         }
48     }
49 }
50 }
```

Questa classe farà uso di due trait...

Metodi esternalizzati nei due trait.

Project Work 14 (facoltativo)

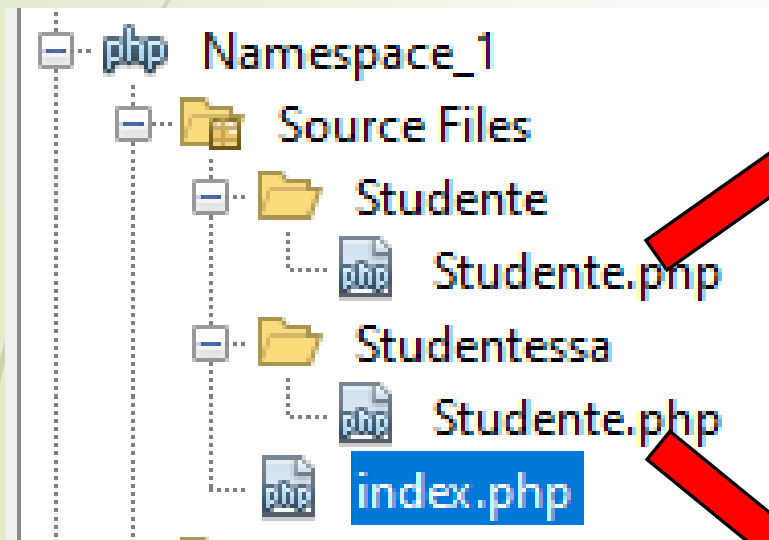
Realizzare un benchmark per testare quanto sia più veloce tenere file/dati in cache APCu, piuttosto che leggerli dal filesystem oppure leggerli da database.



Namespace...

Struttura di un progetto

Consideriamo il seguente progetto che fa uso di classi dentro cartelle separate.



```
13 class Studente {  
14     public function getSaluto() {  
15         echo 'Buon giorno sono uno studente ...';  
16     }  
17 }
```

```
13 class Studente {  
14     public function getSaluto() {  
15         echo 'Buon giorno sono una studentessa ...';  
16     }  
17 }
```

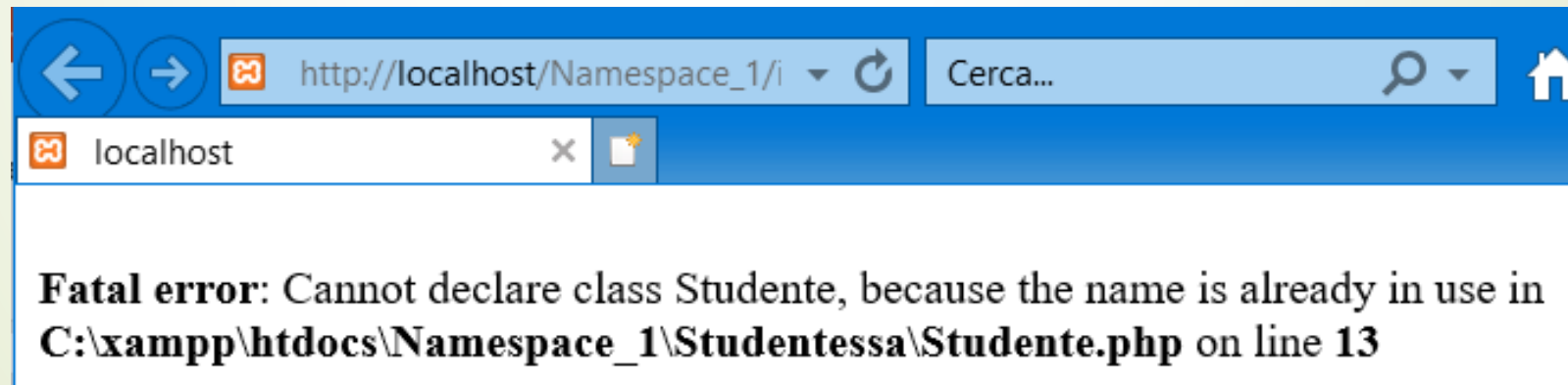
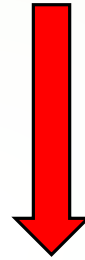
Esempio: «index.php»

Provate ad eseguire questo file... Cosa accade?

```
6  <html>
7  <head>
8      <meta charset="UTF-8">
9      <title></title>
10 </head>
11 <body>
12     <?php
13         include 'Studente/Studente.php';
14         include 'Studentessa/Studente.php';
15
16         $stud = new Studente();
17         echo $stud->getSaluto();
18
19     ?>
20 </body>
21 </html>
```


Namespace: perché usarlo...

Provate ad eseguire questo file... Cosa accade?



Namespace

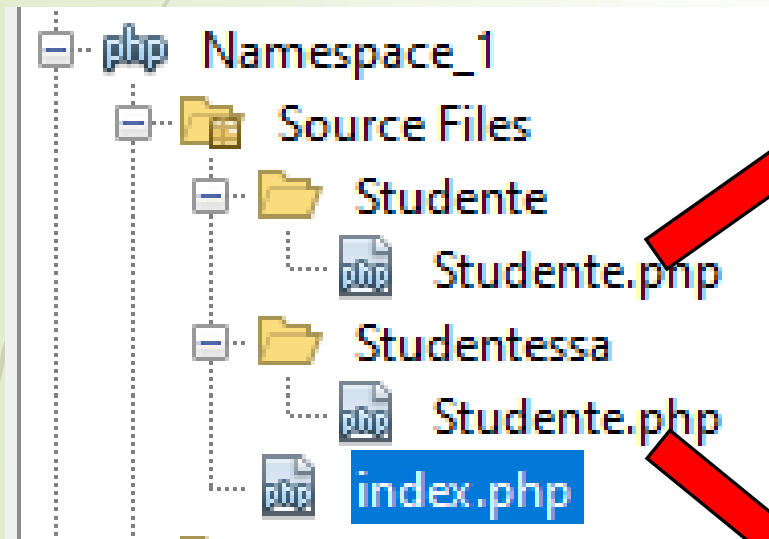
Quando capita di aver bisogno di più librerie, per es. scritte da terze parti, capita di dover gestire omonimie, cioè classi che hanno nomi uguali. Tecnicamente si dice che le librerie vengono usate nello stesso spazio dei nomi, per cui classi con stesso nome causano problemi.

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.

Namespace: come si usa

Quando capita di aver bisogno di più librerie, scritte da terze parti, capita di



```
13 namespace Studente;
14 class Studente {
15     public function getSaluto(){
16         echo 'Buon giorno sono uno studente ...';
17     }
18 }
```

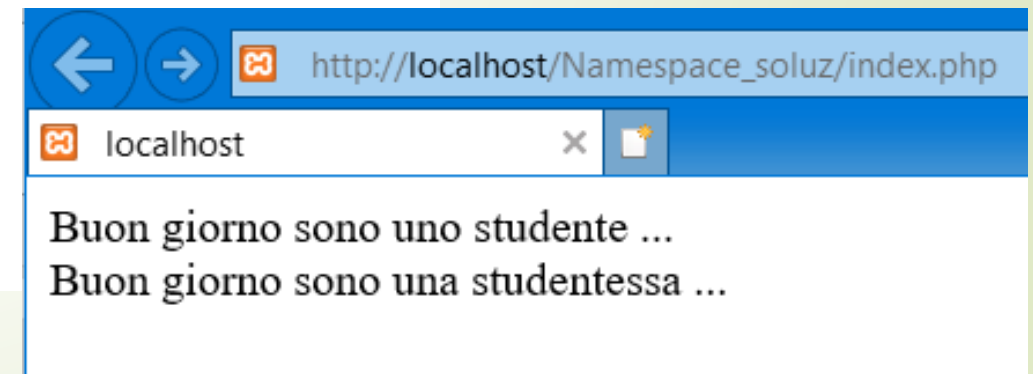
```
13 namespace Studentessa;
14 class Studente {
15     public function getSaluto(){
16         echo 'Buon giorno sono una studentessa ...';
17     }
18 }
```

Namespace: «index.php»

Le istanze tengono conto dei relativi Namespace dichiarati per le classi.

```
11 <body>
12   <?php
13     include 'Studente/Studente.php';
14     include 'Studentessa/Studente.php';
15
16     /*
17      * Quando si istanzia la classe, occorre specificare il Namespace
18      * dove è stata collocata la classe
19      */
20     $studente = new Studente\Studente();
21     $studentessa = new Studentessa\Studente();
22
23     echo $studente->getSaluto()."<br>";
24     echo $studentessa->getSaluto();
25   ?>
26 </body>
```

output





XML...

XML – eXtensible Markup Language

XML è un metalinguaggio utilizzato per definire un grandissimo numero di linguaggi di markup (tra cui, in parte, anche HTML).

Negli anni '90 ci fu il boom di lotte agguerrite tra software house nel settore dei linguaggi e dei browser per il Web... come risposta a questo *farwest*, nel '94 nacque il Consorzio W3C che in quei tempi fu costretto a rincorrere le evoluzioni «de facto» dell'HTML...

Nel '96 all'interno del W3C si costituì l'XML working group che aveva originariamente il compito di definire un linguaggio di markup per il Web che offrisse maggiore libertà nella definizione di tag. Nel '97 il WG pubblicò le prime recommendation per XML.

Presto ci si accorse che XML andava oltre gli obiettivi iniziali, di linguaggio flessibile per il Web, cioè era sufficientemente generale per soddisfare diversi campi d'applicazione. Infatti, fu impiegato per lo scambio di informazioni tra sistemi diversi, per la definizione di formati di dati, per la rappresentazione di immagini, ecc.

XML per HTML

Il linguaggio HTML ha ottenuto una rivisitazione in termini di XML, evoluzione che ha portato alla nascita del **XHTML – eXtensible HTML**. Cioè l'HTML è stato ridefinito secondo le regole sintattiche XML, alcune delle quali sono:

- **tutti i tag e i loro attributi sono espressi in minuscolo**
- **è obbligatorio inserire il tag di chiusura (ad esempio, se usiamo <p> dobbiamo chiudere con </p>)**
- **i valori degli attributi devono essere specificati tra doppi apici o singoli apici (ad esempio, <table width="30%">)**
- **i tag vuoti seguono la cosiddetta sintassi minimizzata (per esempio, il tag
 diventa
)**
- **utilizzare l'attributo id al posto di name per identificare gli elementi di un documento**

Queste regole sintattiche consentono a specifici software di analizzare automaticamente i contenuti Web strutturati in XHTML

XML: la libreria «simpleXML»

Nell'esempio si utilizzano diverse modalità di lettura di contenuti XML.

```
11 <body>
12 <?php
13 include 'HandleXML.php';
14
15 $fileXML = 'phpflow.xml';
16 $fileXML_1 = 'phpflow_1.xml';
17 /*
18  * Accesso tramite file XML
19  */
20 $xmlTest = new HandleXML();
21 $obj_xml_file = $xmlTest->readXmlFile($fileXML);
22 echo "Struttura del file XML: <br>";
23 print_r($obj_xml_file);
24 /*
25  * Accesso tramite stringa XML
26  * Notate la differenza dell'output con l'uso del tag <pre>
27  */
28 $xml_string = $xmlTest->stringXML();
29 echo "<br><br>Struttura della stringa XML: <br><pre>";
30 print_r($xml_string);
31 /*
32  * Accesso tramite "new SimpleXMLElement($string)"
33  */
34 $obj_xml_string = $xmlTest->stringXML();
35 echo "<br><br>Struttura XML usando 'new SimpleXMLElement()': <br>";
36 print_r($obj_xml_string);
37
38 ?>
39 </body>
```

Tre modalità di caricare contenuti XML, gestite dalla classe «[HandleXML.php](#)»

Esempio XML: «HandleXML.php»

```
16 class HandleXML {
17     private $fileToCache = 'cache/phpflow.xml';
18     private $checkForNewFile = 'phpflow.xml';
19
20     function __construct() {
21     }
22
23     public function readXmlFile($fileXML) {
24         $xml = simplexml_load_file($fileXML);
25         //print_r($xml); //tutta la struttura
26         return $xml;
27     }
28
29     function stringXML() {
30         $string = <<<XML
31         <?xml version='1.0'?>
32         <catalogo_film>
33             <cd>
34                 <titolo>Matrix Resurrections</titolo>
35                 <attore>Keano Reeves</attore>
36                 <nazione>USA</nazione>
37                 <regista>Lana Wachowski</regista>
38                 <budget>190 M$</budget>
39                 <year>2022</year>
40             </cd>
41         </catalogo_film>
42         XML;
43         $xml = simplexml_load_string($string);
44
45         return $xml;
46     }
47 }
```

Utilizzo della funzione

«**simple_load_file(\$fileXML)**»

Utilizzo della funzione

«**simple_load_string(\$string)**»

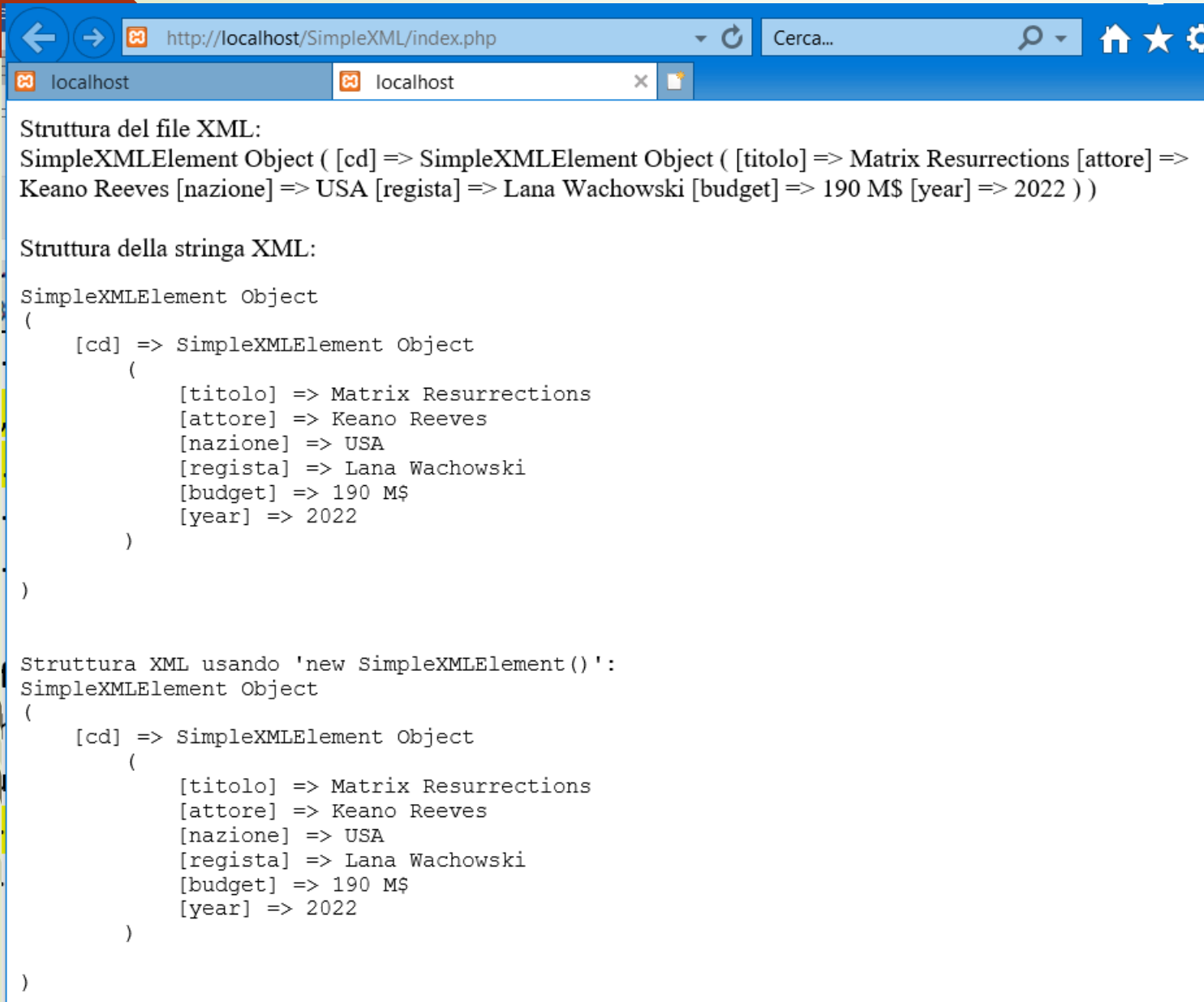
Esempio XML: «HandleXML.php»

```
100 function instanceofXMLObj() {  
101     $string = <<<XML  
102     <?xml version="1.0" encoding="UTF-8"?>  
103     <catalogo_film>  
104         <cd>  
105             <titolo>Matrix Resurrections</titolo>  
106             <attore>Keano Reeves</attore>  
107             <nazione>USA</nazione>  
108             <regista>Lana Wachowski</regista>  
109             <budget>190 M$</budget>  
110             <year>2022</year>  
111         </cd>  
112     </catalogo_film>  
113     XML;  
114     $xml = new SimpleXMLElement($string);  
115  
116     return $xml;  
117 }  
118
```

Frammento di codice della classe
«HandleXML.php»

Utilizzo della funzione
«**SimpleXMLElement(\$string)**»

Esempio XML: output



The screenshot shows a web browser window with the address bar displaying `http://localhost/SimpleXML/index.php`. The page content is as follows:

```
Struttura del file XML:  
SimpleXMLElement Object ( [cd] => SimpleXMLElement Object ( [titolo] => Matrix Resurrections [attore] =>  
Keano Reeves [nazione] => USA [regista] => Lana Wachowski [budget] => 190 M$ [year] => 2022 ) )  
  
Struttura della stringa XML:  
  
SimpleXMLElement Object  
(  
    [cd] => SimpleXMLElement Object  
    (  
        [titolo] => Matrix Resurrections  
        [attore] => Keano Reeves  
        [nazione] => USA  
        [regista] => Lana Wachowski  
        [budget] => 190 M$  
        [year] => 2022  
    )  
)  
  
Struttura XML usando 'new SimpleXMLElement()':  
SimpleXMLElement Object  
(  
    [cd] => SimpleXMLElement Object  
    (  
        [titolo] => Matrix Resurrections  
        [attore] => Keano Reeves  
        [nazione] => USA  
        [regista] => Lana Wachowski  
        [budget] => 190 M$  
        [year] => 2022  
    )  
)
```

Due tipi diversi di formattazione dell'oggetto:

«SimpleXMLElement»

Esempio XML: leggo e stampo

Leggo e stampo tutto il file XML:

«[index_1.php](#)»

```
6 <html>
7   <head>
8     <meta charset="UTF-8">
9     <title></title>
10  </head>
11  <body>
12    <?php
13      include 'HandleXML.php';
14      $fileXML_1 = 'phpflow_1.xml';
15
16      /*
17       * Estrarre elementi particolari
18       */
19      $handle = new HandleXML();
20      $handle->getCDElements($fileXML_1);
21
22      // $handle->getCDSingleElement($fileXML_1, "titolo");
23    ?>
24  </body>
25 </html>
```


Es. XML: «HandleXML.php»

Funzione «[getCDElements\(\\$xmlFile\)](#)» della classe «[HandleXML.php](#)»

```
13 class HandleXML {
14
15     function __construct() { ...2 lines }
16
17
18     public function setCDElements($fileXML_new,$elementi,$catalogo_xml) [
19
20
21
22
23
24
25
26
27
28
29
30
31     public function getCDElements($xmlFile) {
32         $xml = simplexml_load_file($xmlFile);
33         $cd_num = 0;
34
35         foreach ($xml->cd as $singoloCD) {
36             $cd_num++;
37             echo "<br><br>FILM -> CD_". $cd_num. "<br>";
38             echo 'TITOLO: ' . $singoloCD->titolo. "<br>";
39             echo 'Lingua: ' . $singoloCD->lingua. "<br>";
40             echo 'Regista: ' . $singoloCD->regista. "<br>";
41             echo 'Costo: ' . $singoloCD->costo. "<br>";
42             echo 'Anno: ' . $singoloCD->year. "<br>";
43         }
44     }
45 }
```


Es. XML: output

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <catalogo_film>
3   <cd>
4     <titolo>Matrix Resurrections</titolo>
5     <lingua>USA</lingua>
6     <regista>Lana Wachowski</regista>
7     <costo>10€</costo>
8     <year>2022</year>
9   </cd>
10  <cd>
11    <titolo>La vita è bella</titolo>
12    <lingua>Italy</lingua>
13    <regista>Roberto Benigni</regista>
14    <costo>15€</costo>
15    <year>1997</year>
16  </cd>
17 </catalogo_film>
```

http://localhost/SimpleXML/index_1.php

localhost

FILM -> CD_1
TITOLO: Matrix Resurrections
Lingua: USA
Regista: Lana Wachowski
Costo: 10€
Anno: 2022

FILM -> CD_2
TITOLO: La vita è bella
Lingua: Italy
Regista: Roberto Benigni
Costo: 15€
Anno: 1997

Project Work 15

Realizzare un nuovo metodo per la classe «[HandleXML.php](#)», che chiamerete per comodità «[getSingleElement\(\\$xmlFile,\\$ele\)](#)», che produca l'estrazione di un singolo elemento (\$ele) dalla struttura XML.

PW-15: contenuto del file XML



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <catalogo_film>
3   <cd>
4     <titolo>Matrix Resurrections</titolo>
5     <lingua>USA</lingua>
6     <regista>Lana Wachowski</regista>
7     <costo>10€</costo>
8     <year>2022</year>
9   </cd>
10  <cd>
11    <titolo>La vita è bella</titolo>
12    <lingua>Italy</lingua>
13    <regista>Roberto Benigni</regista>
14    <costo>15€</costo>
15    <year>1997</year>
16  </cd>
17 </catalogo_film>
```

PW-15: soluzione proposta

```
47 public function getCDSingleElement($xmlFile,$sele) {
48     $xml = simplexml_load_file($xmlFile);
49     $cd_num = 0;
50
51     foreach ($xml->cd as $singoloCD) {
52         $cd_num++;
53         echo "<br><br>FILM -> CD_". $cd_num. "<br>";
54         switch($sele) {
55             case 'titolo':
56                 echo 'TITOLO: ' . $singoloCD->titolo. "<br>";
57                 break;
58             case 'lingua':
59                 echo 'Lingua: ' . $singoloCD->lingua. "<br>";
60                 break;
61             case 'regista':
62                 echo 'Regista: ' . $singoloCD->regista. "<br>";
63                 break;
64             case 'costo':
65                 echo 'Costo: ' . $singoloCD->costo. "<br>";
66                 break;
67             case 'year':
68                 echo 'Anno: ' . $singoloCD->year. "<br>";
69                 break;
70             default:
71                 echo 'Valore non riconosciuto!';
72         }
73     }
74 }
```

PW-15: «index_1.php» e output

```
12 <?php
13     include 'HandleXML.php';
14     $fileXML_1 = 'phpflow_1.xml';
15
16     /*
17     * Estrarre elementi particolari
18     */
19     $handle = new HandleXML();
20     //$handle->getCDElements($fileXML_1);
21
22     $handle->getCDSingleElement($fileXML_1,"titolo");
23     ?>
```



Creare un file XML: «index_2.php»

```
12 <?php
13 include 'HandleXML.php';
14 $fileXML_new = 'phpflow_new.xml';
15
16 /*
17  * Estrarre elementi particolari
18  */
19 $handle = new HandleXML();
20
21 $catalogo_xml = new SimpleXMLElement('<?xml version="1.0" encoding="UTF-8"?>
22 <catalogo_film></catalogo_film>');
23
24 $elementi = ['titolo', 'lingua', 'regista', 'costo', 'year'];
25 $handle->setCDElements($fileXML_new, $elementi, $catalogo_xml);
26
27 $elementi = ['titolo1', 'lingual', 'registal', 'costol', 'year1'];
28 $handle->setCDElements($fileXML_new, $elementi, $catalogo_xml);
29
30 //$handle->getCDElements($fileXML_1);
31 ?>
```

Creare un file XML: «HandleXML.php»

```
13 class HandleXML {
14
15     function __construct() {...2 lines }
16
17
18     public function setCDElements($fileXML_new,$selementi,$catalogo_xml){
19         $cd_ele = $catalogo_xml->addChild('cd');
20
21         $cd_ele->addChild('titolo',$selementi[0]);
22         $cd_ele->addChild('lingua',$selementi[1]);
23         $cd_ele->addChild('regista',$selementi[2]);
24         $cd_ele->addChild('costo',$selementi[3]);
25         $cd_ele->addChild('year',$selementi[4]);
26
27         $catalogo_xml->saveXML($fileXML_new);
28         echo 'File xml salvato: '.$fileXML_new;
29     }
30
31     public function getCDElements($xmlFile) {...14 lines }
32
33
34     public function getCDSingleElement($xmlFile,$ele) {...28 lines }
35
36
37     public function readXmlFile($fileXML){
38         $xml = simplexml_load_file($fileXML);
39         //print_r($xml);//tutta la struttura
40         return $xml;
41     }
42
43
44     function stringXML() {...18 lines }
45
46
47     function instanceOfXMLObj() {...18 lines }
48
49 }
```




GRAZIE!