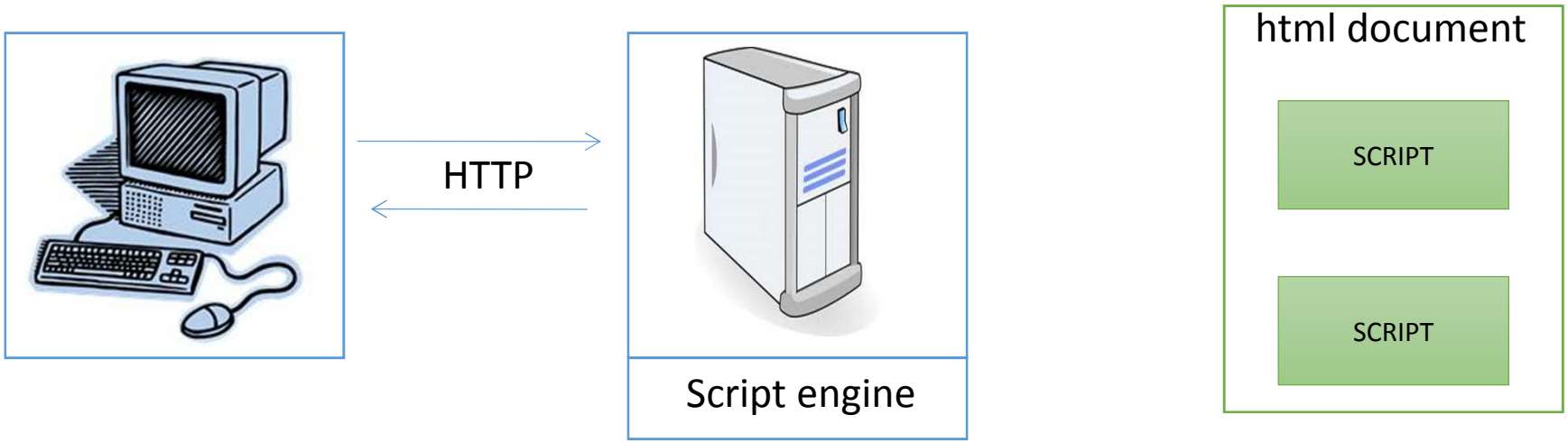
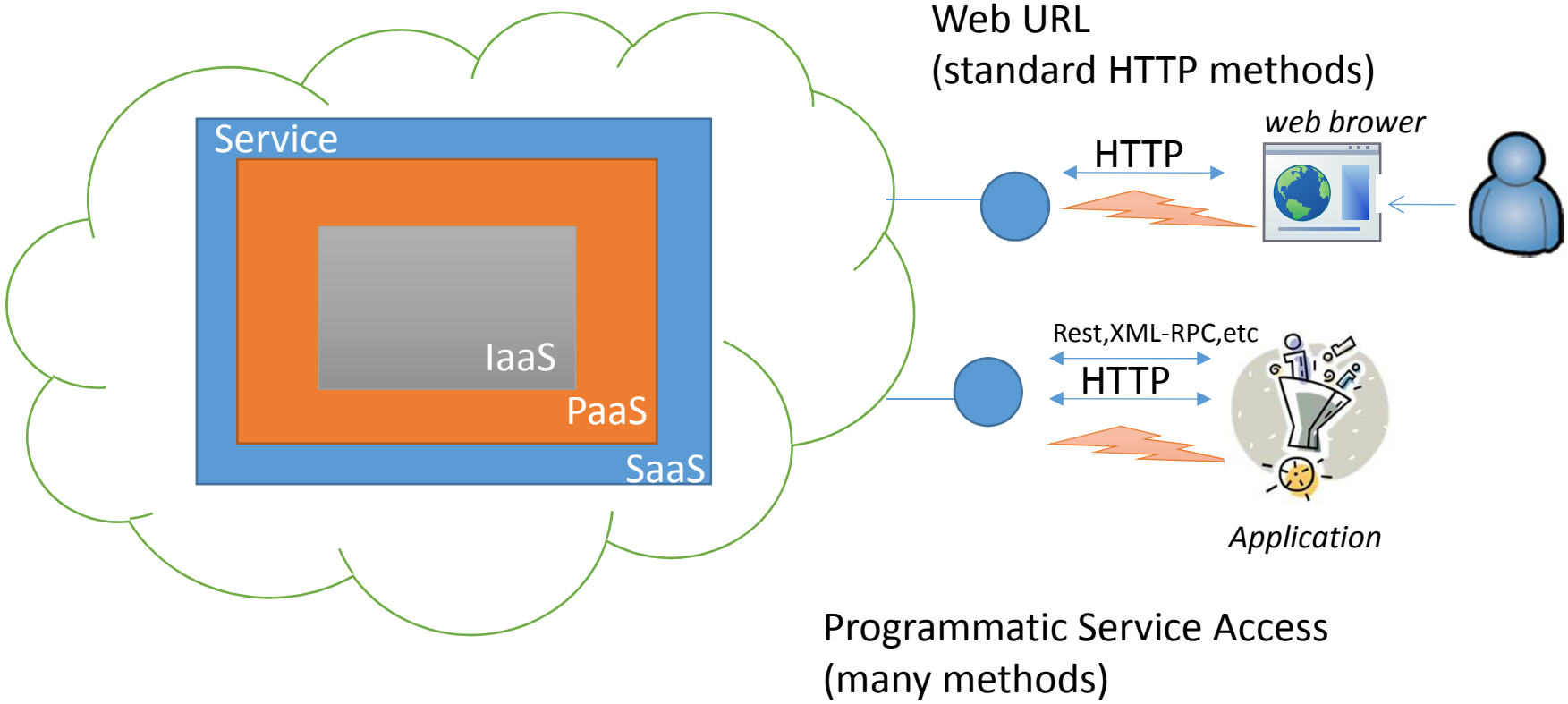
An Introduction to Php for Web API

Principle of server side script



- Pages are generated by a program
- A html document at the server side includes the code to be executed (script)
- The code is delimited via special escape characters
- The web server extracts the script part from the document
- A script engine runs the code
- Web server replaces the script with the output of the execution
- Client sees pure html (no way to access the code)

Access to Cloud computing



- A service is delivered through two access points:
- Standard web browsing (HTTP) and
- Programmatic access (Rest,XML-RPC,SOAP, etc. over HTTP)

Introduction to PHP

- Scripting language
- Server side execution
  - Code is scattered inside a html document
  - The web server executes the code and produces a simple html page
- Useful to implement a service in the cloud
  - Other technologies are available

PHP code embedding

```
<HTML>
<HEAD>Sample PHP
  Script</HEAD>
<BODY>
The following prints "Hello, World":
<?php
  print "Hello, World";
?>
</BODY>
</HTML>
```

Every time the PHP interpreter reaches a PHP open tag **<?php**,it runs the enclosed code up to the delimiting **?>**marker.

Can be changed, see `short_open_tags` INI option;

PHP code embedding

```
<HTML>
<HEAD>Sample PHP
  Script</HEAD>
<BODY>
The following prints "Hello, World":
<?php
  print "Hello, World"
?>
</BODY>
</HTML>
```



```
<HTML>
<HEAD>Sample PHP Script</HEAD>
<BODY>
The following prints "Hello, World":
Hello, World
</BODY>
</HTML>
```

Every time the PHP interpreter reaches a PHP open tag <?php,it runs the enclosed code up to the delimiting ?>marker.

Variables

- A variable always starts with the **dollar sign** $
  - $a
  - $A
  - $1 (not allowed)
- Identifiers are case sensitive (not when referring to function)
- Variable and function can have the same name!

Types

- Basic types like in other programming languages
  - Boolean, Integer, Floating Point, Object,
- Main difference concerns:
  - string (regular expression,…)
    - single quoted (variables are not replaced with their values)
    - double quoted (variables are replaced with their values)
    - …
  - array (associative arrays)
- Other types:
  - null
    - No type associated yet
  - resource
    - Generic type, e.g. the result of a query

Types

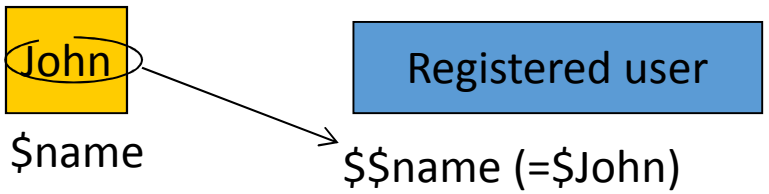
- PHP uses a Weakly Typed System
- variables' type is not declared
- PHP automatically converts the variable to the correct data type, depending on how they are set
- $integer=10
- $float = 10.0
- $string = "10"

Some example

```
$a = “fine” // $a is a string
$a = 10;     // $a is an integer
$b = 6.3;
$c = $a + $b; /* $c is a float */
$d = (int)$c; // type casting ($d integer)

gettype($d);
settype($d, double); // $d is now double

print(gettype($e)); // print boolean

if (is_int($d))  // is_type to type check
```

Variable variables

```
<?php
$name = "John";
$$name = "Registered user";
print $John; //display “Registered user”
?>
```



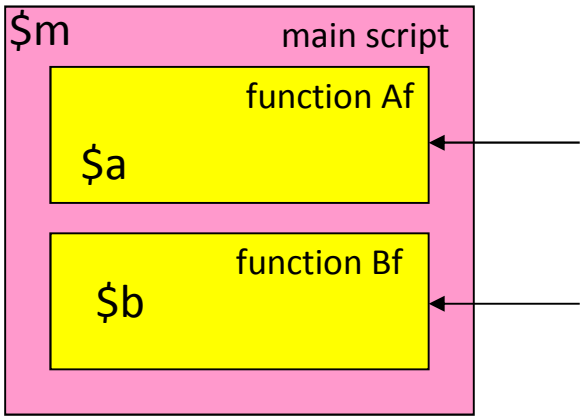
Managing variables

- isset ()
  - determines whether a certain variable has already been declared by PHP.
- unset()
  - “undeclares” a previously set variable, and frees any memory that was used by it if no other variable references its value.

- empty ()
  - empty() may be used to check if a variable has not been declared or its value is false.

Variable's scope

- Names inside a function has local scope
- Script level names can be accessed through the special built-in array **$GLOBALS**



$a is only visible in the function Af's scope
$m can be seen via $GLOBALS[m]

$b is only visible in the function Bf's scope
$a is not visible
$m can be seen via $GLOBALS[m]

Predefined System "Superglobals"

- Provide access to key runtime data elements.
- Set by and managed through web server run-time environment and available to the script.
- Superglobals are key to form processing, cookies, and other techniques.

Some Superglobals

- $_GET[ ]. An array that includes all the GET variables that PHP received from the client browser.
- $_POST[ ]. An array that includes all the POST variables that PHP received from the client browser.
- $_COOKIE[ ]. An array that includes all the cookies that PHP received from the client browser.
- $_SERVER[ ]. An array with the values of the web-server variables.
- $_SESSION[ ]. Array with values concerning a ‘session’

Output: echo statement

- Placing a variable outside quotes outputs the variable's value (line 2)
- Single quote ' sends literal string output (line 3), no variable value substitution
- Double quote " sends variable value (line 4)

```
<?php
1  $a=6;
2  echo $a;
3  echo 'The var name is $a';
4  echo "The var contains $a";
?>
```

Note: no declaration (line 1)

Constant

- Unchangeable values. In all caps by convention. No $.

```
<?php
define('MYCONST',100);
define('NAME',"My Name");
?>
```

- To output, must list constant name outside of ' and ".
- **echo "Hello, ".NAME;**
- Predefined system constants also exist.
- To see a complete list:

  print_r(get_defined_constants())

Output: print_r()

- **print_r()** can be used to "dump" variable output, typically for debugging of complex structures.

```
<?php
print_r($_SERVER);
?>
```

Example

```
<?php
        $user = (isset($_GET[‘user’]) ? $_GET[‘user’]:”” );
…
?>
```

Comments

- Multi-line comments

```
/* This is a multi-line comment */
```

- Single line comments

```
// This single line is commented

# So is this single line
```

  - PHP comments are distinct from HTML comments in that PHP comments are not sent to the client browser.

Operators

- **+, -, *, /, %, ++, --** same as other languages
- Combining above with = for assignment can be done:
- **+=, -=, *=, /=, %=, .=**
- Two Comparison operators
- == (performs type conversion)
- === (no type conversion)
- ‘1’==1 → true
- ‘1’===1 →false

Creating a form

- Key elements:
  - Input fields must be contained inside a **form tag.**
  - All input fields must have a **name.**
  - Names cannot have spaces in them. Fields should be named well for clear identification.
- Form **action** should be URL to PHP processing script.
- Appropriate form transmission method selected:
  - **GET** or **POST.**

GET vs POST

- *Name/value* pairs appended in clear text to the URL of the receiving page/script.
- Each name/value pair separated by '&'. Value data automatically URL encoded.
- Names are taken from the form field names.
- GET URLs can be saved, bookmarked, etc. and used to recall the script with the same data.
- GET strings provide 'transparency' that may/may not be desired.
- Data available into the $_GET superglobal

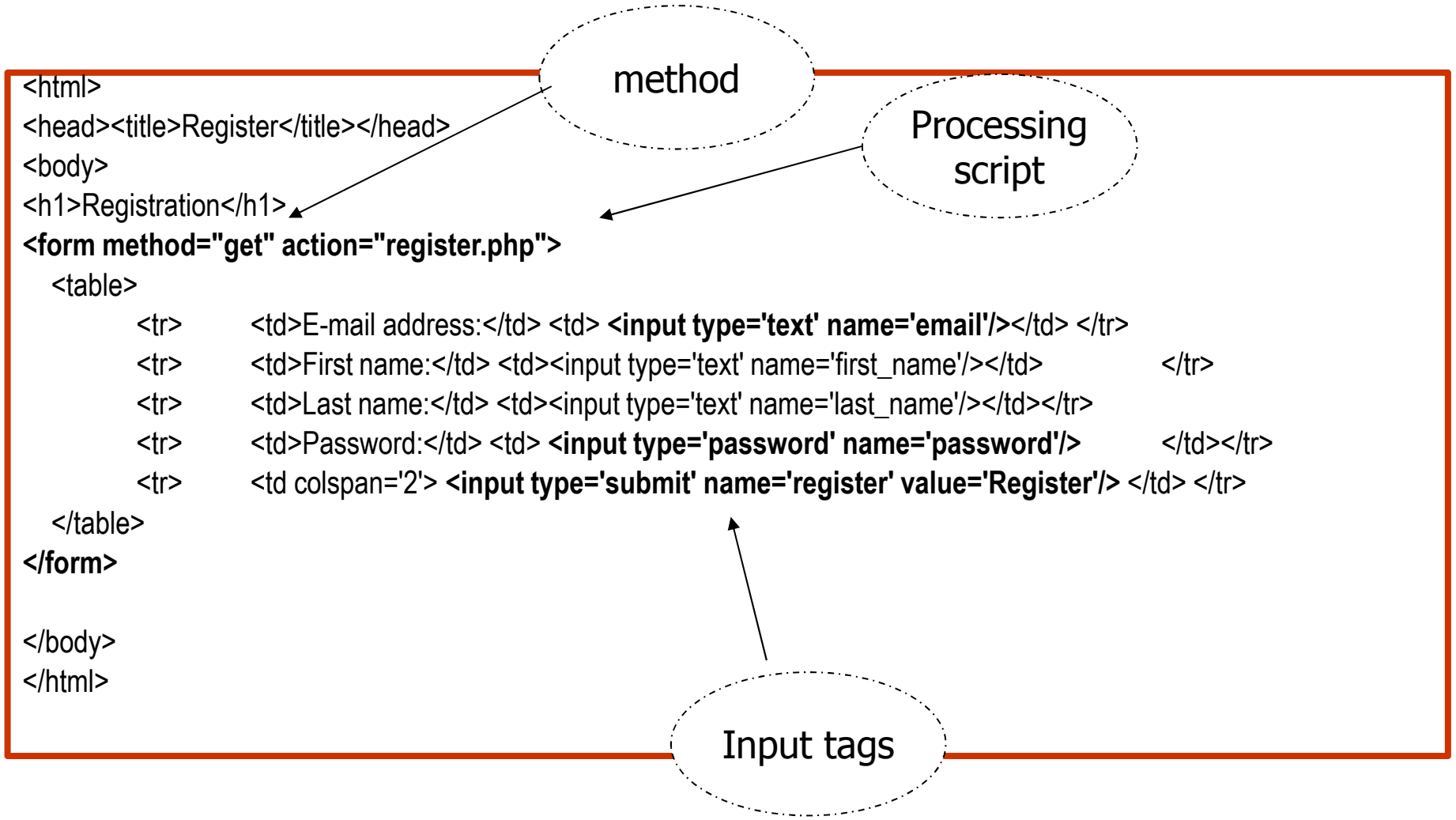
GET vs *POST*

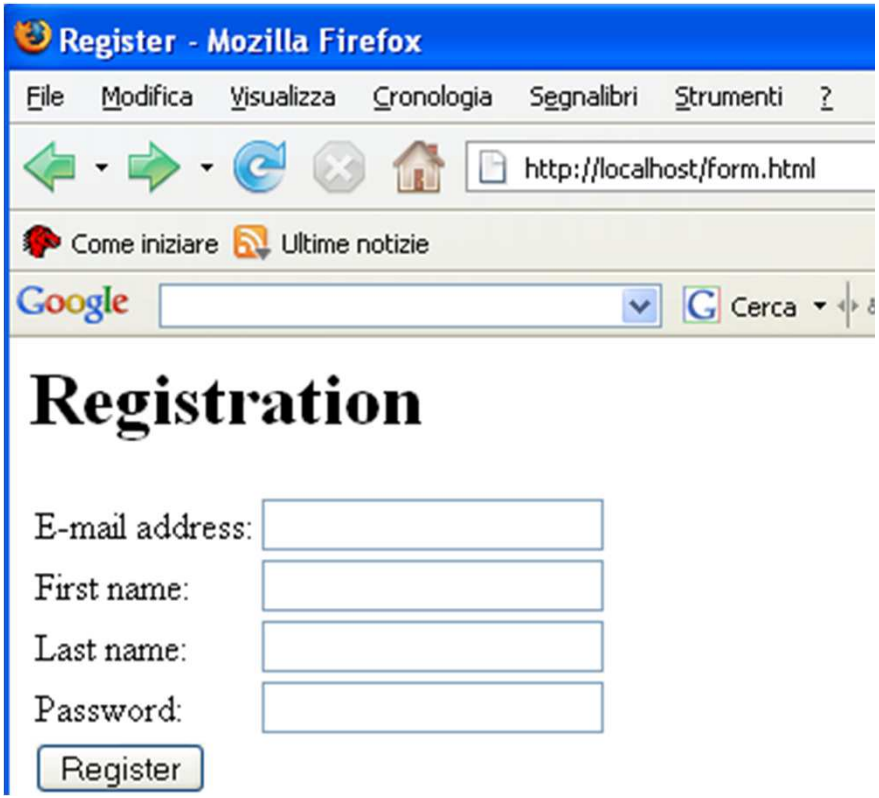
- Data is encoded in the page request body sent by the browser, but not shown in the URL. Unseen to user.
- Since data not part of URL, bookmarking and reusing URL to recall the script with the same data is not possible.
- Large POST packets not a problem.
- Data available into the $_POST superglobal

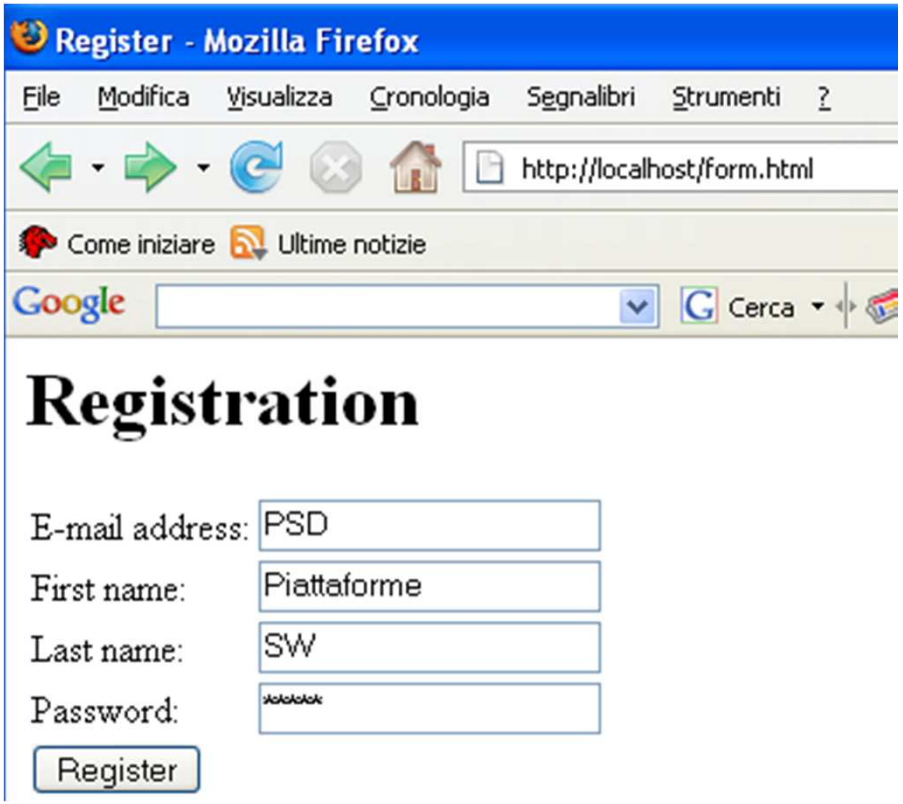
An example

```
<html>
<head><title>Register</title></head>
<body>
<h1>Registration</h1>
<form method="get" action="register.php">
   <table>
            <tr>      <td>E-mail address:</td> <td> <input type='text' name='email'/></td> </tr>
            <tr>      <td>First name:</td> <td><input type='text' name='first_name'/></td>          </tr>
            <tr>      <td>Last name:</td> <td><input type='text' name='last_name'/></td></tr>
            <tr>      <td>Password:</td> <td> <input type='password' name='password'/>          </td></tr>
            <tr>      <td colspan='2'> <input type='submit' name='register' value='Register'/> </td> </tr>
   </table>
</form>

</body>
</html>
```

An example


method
Processing script
<html>
<head><title>Register</title></head>
<body>
<h1>Registration</h1>
<form method="get" action="register.php">
<table>
<tr> <td>E-mail address:</td> <td> <input type='text' name='email'/></td> </tr>
<tr> <td>First name:</td> <td><input type='text' name='first_name'/></td> </tr>
<tr> <td>Last name:</td> <td><input type='text' name='last_name'/></td></tr>
<tr> <td>Password:</td> <td> <input type='password' name='password'/> </td></tr>
<tr> <td colspan='2'> <input type='submit' name='register' value='Register'/> </td> </tr>
</table>
</form>
</body>
</html>
Input tags





key value

http://localhost/register.php?email=PSD&first_name=Piattaforme&last_name=SW&password=Pippo&register=Register

Conditional control structures

```
if (expr)
   statement
elseif (expr)
   statement
elseif (expr)
   statement
...
else
   statement
```

```
{ statement1;
  statement 2;
 }
```

```
if (expr):
      statement list
elseif (expr) :
      statement list
...
else :
      statement list
endif;
```

Traditional loop control structures

```
while (expr)
    statement
```

```
while (expr) :
        statement list
endwhile;
```

```
do
        statement
while (expr);
```

```
for (expr, expr, …; expr, expr, …; expr, expr, …)
 statement
```

```
for ($i = 0; $i <= count($array); $i++) {
}
```

```
$count = count($array);
for ($i = 0; $i <= $count; $i++) {
}
```

Array

array([key =>] value, [key =>] value, ...)

- The key is optional, and when it's not specified, the key is automatically assigned one more than the largest previous integer key (starting with 0).

- There are three different kind of arrays:
  - Numeric array - An array with a numeric ID key
  - Associative array - An array where each ID key is associated with a value
  - Multidimensional array - An array containing one or more arrays

Examples

1. array(1, 2, 3)
2. array(0 => 1, 1 => 2, 2 => 3)
3. array ("name" => "John", "age" => 28)
4. array(1 => "ONE", "TWO", "THREE")
5. array(1 => "ONE", 2 =>"TWO", 3 => "THREE")
6. array (array ("name" => "John", "age" => 28), array ("name" => "Barbara", "age" => 67))

1 and 2 are same, 4 and 5 are same, 6 is a nested array

Examples

```
$arr1 = array(1, 2, 3);
$arr2[0] = 1;
$arr2[1] = 2;
$arr2[2] = 3;
```

print_r($arr1)

```
Array
(
[0] => 1
[1] => 2
[2] => 3
)
```

```
$arr1 = array("name" => "John", "age" =>28);
$arr2["name"] = "John";
$arr2["age"] = 28;
if ($arr1 == $arr2) {
print '$arr1 and $arr2 are the same';
}
```

```
$arr1 and $arr2 are the same
```

Traversing

foreach($array as [$key =>] [&] $value)

- $key contains the currently iterated value's key
- & if present allows to modify the array
- $value contains the value

```
$players = array ("John", "Barbara", "Bill", "Nancy");
print "The players are:<br>";
foreach ($players as $key => $value) {
print "#$key = $value<br>";
}
```

The players are:
#0 = John
#1 = Barbara
#2 = Bill
#3 = Nancy

Array related functions

^Array Functions

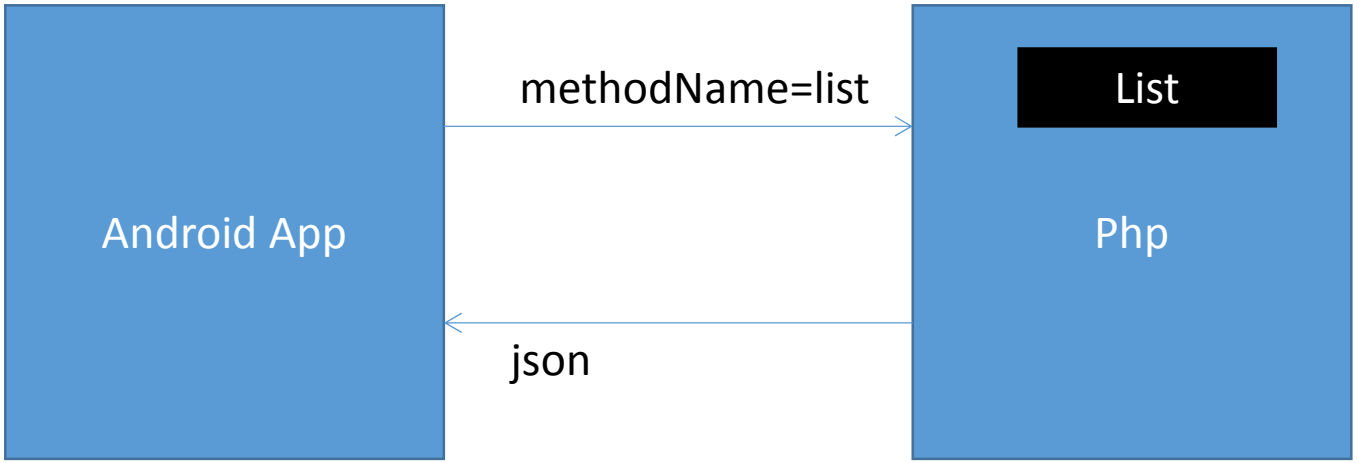
- array_change_key_case
- array_chunk
- array_combine
- array_count_values
- array_diff_assoc
- array_diff_key
- array_diff_uassoc
- array_diff_ukey
- array_diff
- array_fill_keys
- array_fill
- array_filter
- array_flip
- array_intersect_assoc
- array_intersect_key
- array_intersect_uassoc
- array_intersect_ukey
- array_intersect
- array_key_exists
- array_keys
- array_map
- array_merge_recursive
- array_merge
- array_multisort
- array_pad
- array_pop

- array_walk
- array
- arsort
- asort
- compact
- count
- current
- each
- end
- extract
- in_array
- key
- krsort
- ksort
- list
- natcasesort
- natsort
- next
- pos
- prev
- range
- reset
- rsort
- shuffle
- sizeof
- sort
- uasort
- uksort
- usort

...

json_encode

Lab: simpe Shopping cart



Add Items to the server from a web form (server side)
List item in the telephone
Add item into a DB on the server

Example

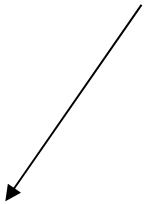
```php
<?php
function cube($n)
{
    return($n * $n * $n);
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cube", $a);
print_r($b);
?>
```

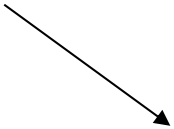
Functions

- Any valid PHP code may appear inside a user-defined function, even other function…
- Functions need not be defined before they are referenced
- Call-by-reference, call-by-value, default value, variable-length argument, lambda-style function

Parameter passing

function *function_name* (*arg1*, *arg2*, *arg3*, …)
{
statement list
}





```
function square(&$n)
{
$n = $n*$n;
}
```

… by-reference

parameter by-value

```
function square($n)
{
$n = $n*$n;
}
```

Default value

```
function makeAcoffee ($type="espresso")
{
 return "Making a cup of $type";
}
```

```
echo makeAcoffee();
echo MakeAcoffee("French")
```

- The default value must be a constant
- Default arguments should be on the right side of any non-default argument

Variable-length argument list

```
function foo()
   {
      $numargs = func_num_args();
      echo "Number of arguments: $numargs\n";
    }


   foo(1, 2, 3);
```

Variable function

- If a variable name has parentheses appended to it, PHP looks for a function with that name and executes it

```
function foo() {echo “in foo()<br>”;}


$func = ‘foo’;
$func(); #call foo()
```

Static variables

```
function do_something()
{
static $first_time = true;
if ($first_time) {
// Execute this code only the first time the function is called
...
$first_time=false;
}
// Execute the function's main logic every time the function is called
...
}
```

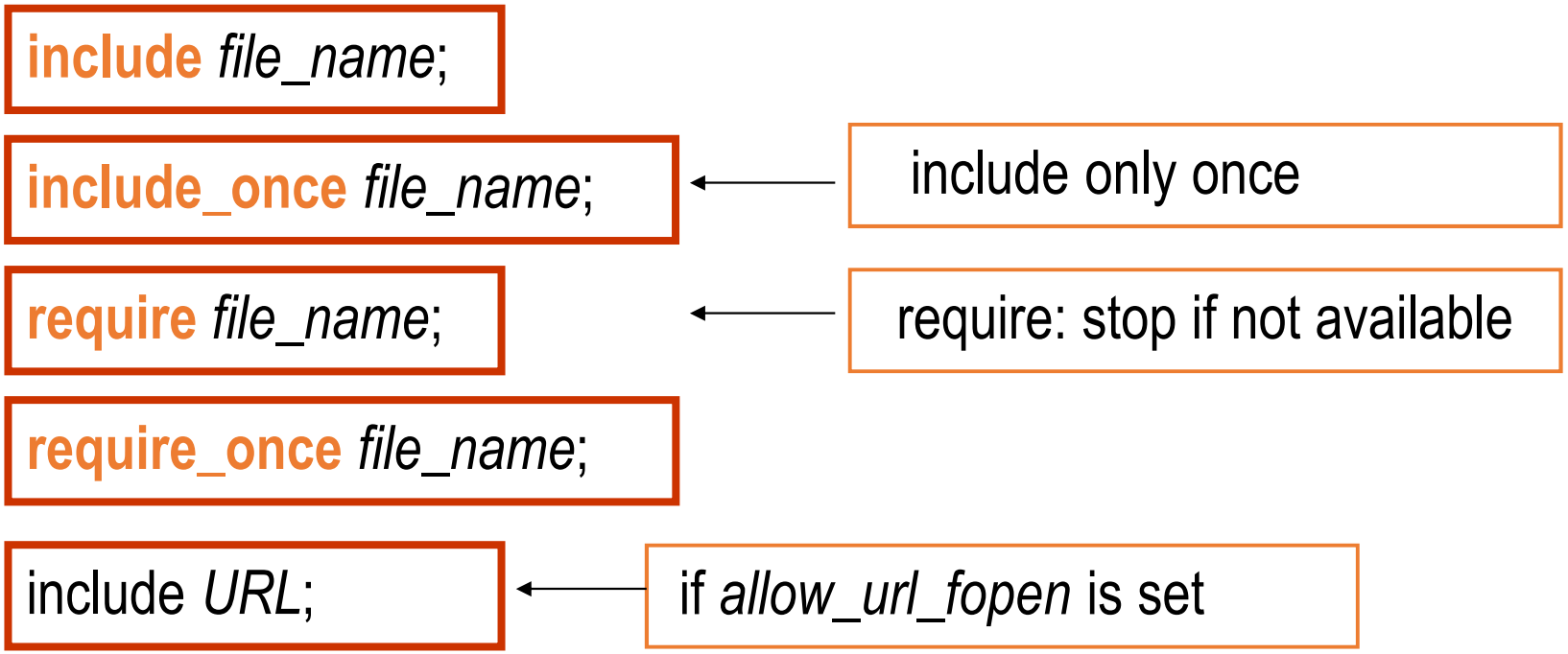
Array_map

- Applies a *callback* function to the elements of the given arrays

```php
<?php
function Double($a){return 2*$a;};

$in = range(1,5);

$out = array_map("Double",$in);

print_r($out);
?>
```

- Other interesting functions (see manual):
- **array_walk**
- **array_reduce**
- …

Code inclusion control structures

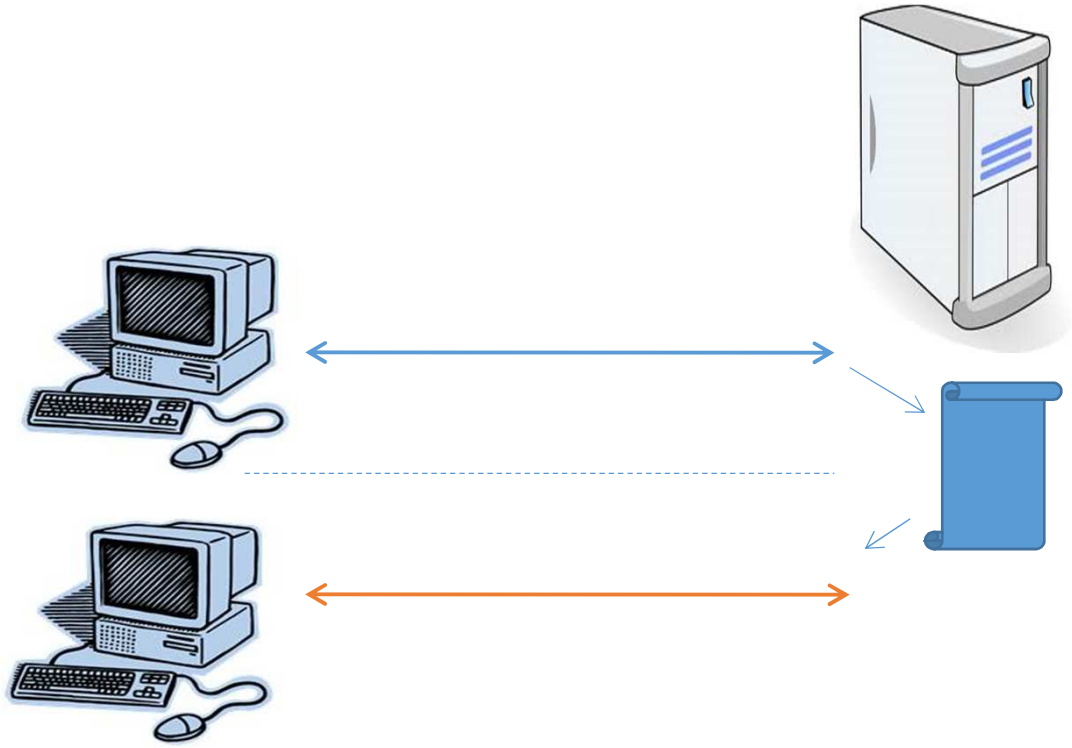



```
include "http://www.example.org/example.php";
```

```
include $_SERVER["DOCUMENT_ROOT"] . "/myscript.php";
```

Persistency

- Cookie, Session
  - Per browser data storage, no cross-browser data exchange
- File, DB
  - Site level persistence storage



SQLite / SQLite3

- Light and fast, shipped with php5
- No dedicated servers are required
- Procedural and object oriented APIs
- Cons: Lock mechanism is not very efficient

Methods

- query - Execute a query
- queryExec - Execute a result-less query
- arrayQuery - Execute a query and return the result as an array
- singleQuery - Execute a query and return either an array for one single column or the value of the first row
- unbufferedQuery - Execute an unbuffered query
- lastInsertRowid - Returns the rowid of the most recently inserted row
- changes - Returns the number of rows changed by the most recent statement
- createAggregate - Register an aggregating UDF for use in SQL statements
- createFunction - Register a UDF for use in SQL statements
- busyTimeout - Sets or disables busy timeout duration
- lastError - Returns the last error code of the most recently encountered error
- fetchColumnTypes - Return an array of column types from a particular table

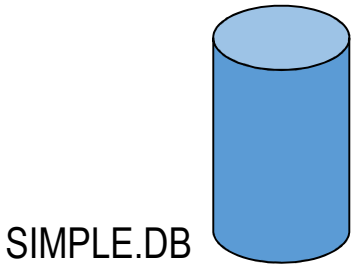
Open/create a db

resource **sqlite_open** (string $filename [, int $mode [, string &$error_message]])

Opens a SQLite database or creates the database if it does not exist.

```
<?php
if ($db = sqlite_open("SIMPLE.DB",0666,&$error))
   print("DB OPENED...."."\n");
 else
    die($error);
?>
```

```
<?php
$db = new SQLiteDatabase("SIMPLE.DB", &$error);
if ($db)
   echo "DB OPENED....";
 else
    die($error);
?>
```



Create a table

bool **queryExec** (string $query [, string &$error_msg])

Executes a result-less query against a given database

Sql statment →

Two types: integer and text →

```
$create_query = "
  CREATE TABLE PRODUCTS (
  id integer primary key,
  description,
  quantity integer
 )
";
 $db->queryExec($create_query);
```

db

PRODUCTS

id	description	quantity

Insert a row

```
$query = "INSERT INTO PRODUCTS (id,description,quantity) VALUES (1,'DVD',1)";
$db->queryExec($query);
```

id	Description	quantity
1	DVD	1

PRODUCTS

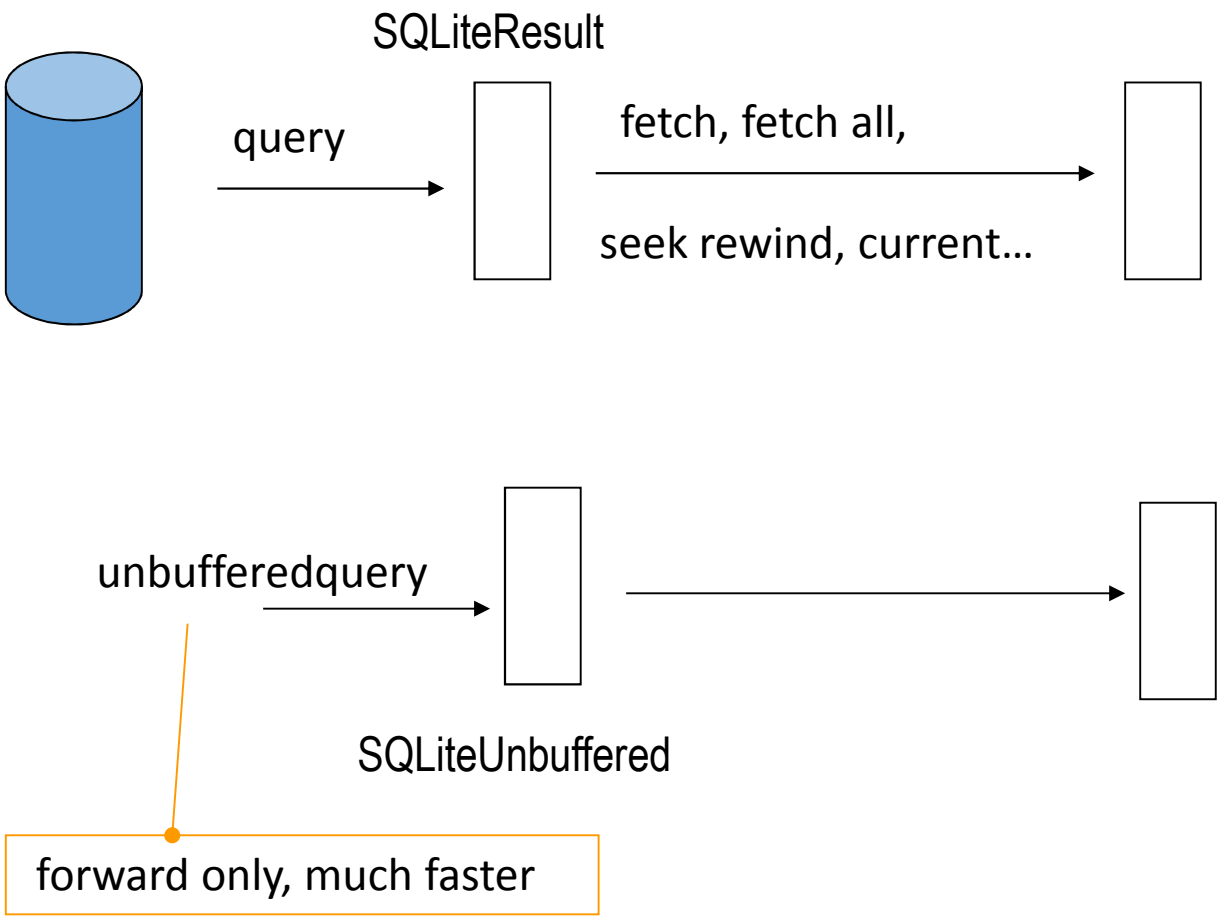
db

Update/delete

```
$db->queryexec('DELETE FROM PRODUCTS WHERE id=2');
```

```
$db->queryexec('UPDATE PRODUCTS SET id=19 WHERE id=4');
```

Fetch results



Fetch results

```
$q = "SELECT * FROM PRODUCTS;";
$qr = $db->query($q); //Executes a query against a given database and returns a result handle
$r = $qr->fetchAll();//Fetches all rows from a result set as an array of arrays

foreach ($r as $entry)   {
echo $entry['id'].' '.$entry['description'].' '.$entry['quantity'].'<br>';
}
```

1 DVD 1

OO Model

- An OO program is a collection of objects
- Every object is an instance of a class
- An object has properties
- An object has a set of methods

Constructor

- Unified constructor name
- `__construct()`

```
class MyClass {
function __construct() {
echo "Inside constructor";
}
}
```

Destructor

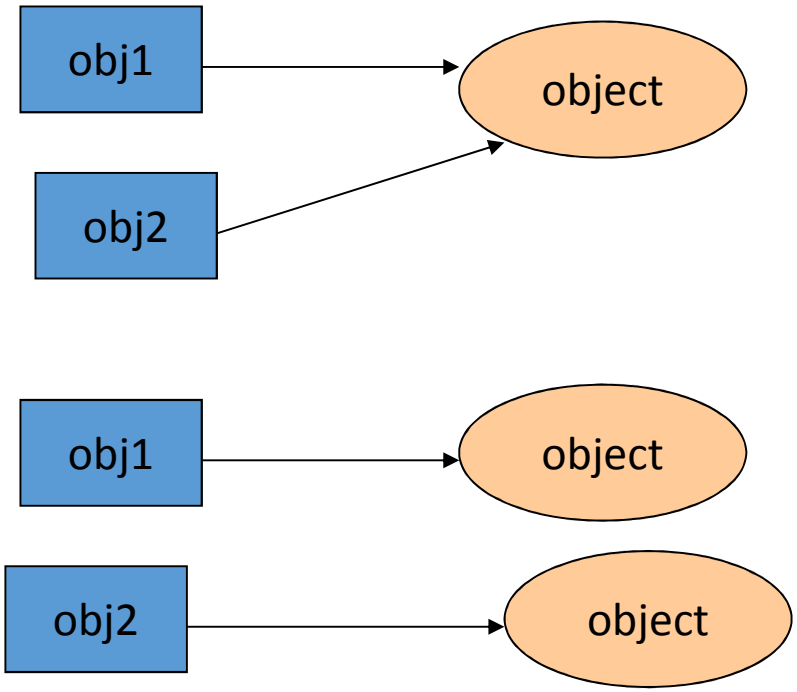
- `__destruct()`
- Called when an object is destroyed (no more reference)

```
class MyClass {
function __destruct()
{
print "An object of type MyClass is being
destroyed\n";
}
}
```

```
$obj = new MyClass();
$obj = NULL;
```

An object of type MyClass is being destroyed

Copying objects




```
class MyClass {
public $var = 1;
}
```

```
$obj1 = new MyClass();
$obj2 = $obj1;
$obj2->var = 2;
print $obj1->var; //print 2
```

```
$obj1 = new MyClass();
$obj2 = clone $obj1;
$obj2->var = 2;
print $obj1->var; //print 1
```

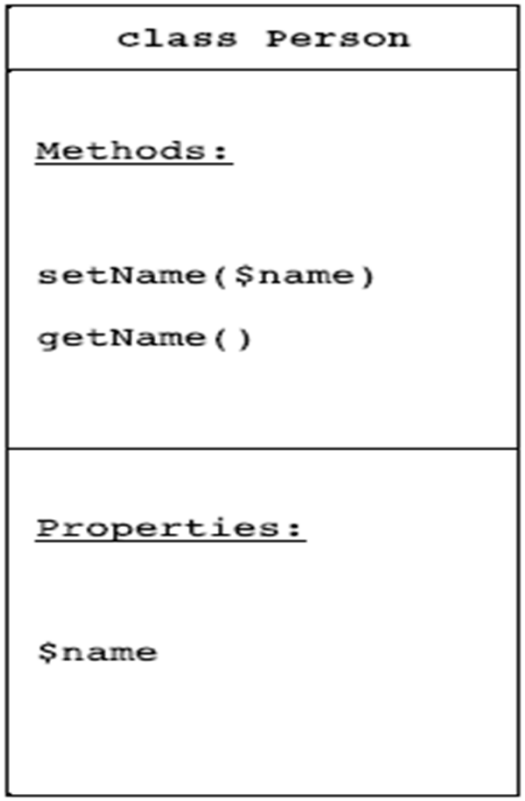
Access protection of member variables

```
class MyDbConnectionClass {
public $queryResult;
protected $dbHostname = "localhost";
private $connectionHandle;
// ...
}
```



```
class MyFooDotComDbConnectionClass extends MyDbConnectionClass {
protected $dbHostname = "foo.com";
}
```

Example




```
class Person {
private $name;

function setName($name)
{
$this->name = $name;
}

function getName()
{
return $this->name;
}
};
```

```
$judy = new Person();
$judy->setName("Judy");
$joe = new Person();
$joe->setName("Joe");
print $judy->getName() . "\n"; //print Judy
print $joe->getName(). "\n"; //print Joe
```

Static properties

self: refer to the current class

```
class MyUniqueIdClass {

static $idCounter = 0;
public $uniqueId;
function __construct()
{
self::$idCounter++;
$this->uniqueId = self::$idCounter;
}
}
```

```
$obj1 = new MyUniqueIdClass();
print $obj1->uniqueId ; //print 1
$obj2 = new MyUniqueIdClass();
print $obj2->uniqueId ; //print 2
```

POLYMORPHISM

- Single class inheritance
  - like Java
- Multiple interface implementations
  - Final keyword

```
class Child extends Parent {
...
}
```

```
class A implements B, C, ... {
...
}
```

```
interface I1 extends I2, I3, ... {
...
}
```

```php
<?php
class Auth {
 function Auth()
{
 mysql_connect('localhost', 'user', 'password');
 mysql_select_db('my_own_bookshop');
}
```

```php
public function addUser($email, $password)
{
$q = '
INSERT INTO users(email, passwd)
VALUES (“ '. $email. ‘ ", “ '. sha1($password).‘ ")
';
mysql_query($q);
}
```

```
public function authUser($email, $password)
{
$q = '
SELECT * FROM users
WHERE email=“ '. $email. ' "
AND passwd =“ '. sha1($password). ' "
';
$r = mysql_query($q);
if (mysql_num_rows($r) == 1) {
return TRUE;
} else {
return FALSE;
} } }
?>
```

Reflection

- Allows to have class information at run-time
- Just an example

```php
<?php

class C {
function F()
{
print "Hello, World\n";
}
}

ReflectionClass::export("C");
?>
```



```
...
 - Constants [0] { }
 - Static properties [0] { }
 - Static methods [0] { }
 - Properties [0] { }
 - Methods [1] {
                Method [  public method F ]
 ...
```

PHP Communication

string **file_get_contents** (string $filename [...])

Reads entire file into a string

```
<?php
/* Identical to above, explicitly naming FILE scheme */
$localfile = file_get_contents("file:///home/bar/foo.txt");

/* Read remote file from www.example.com using HTTP */
$httpfile  = file_get_contents("http://www.example.com/foo.txt");

/* Read remote file from www.example.com using HTTPS */
$httpsfile = file_get_contents("https://www.example.com/foo.txt");

/* Read remote file from ftp.example.com using FTP */
$ftpfile   = file_get_contents("ftp://user:pass@ftp.example.com/foo.txt");

/* Read remote file from ftp.example.com using FTPS */
$ftpsfile  = file_get_contents("ftps://user:pass@ftp.example.com/foo.txt");
?>
```

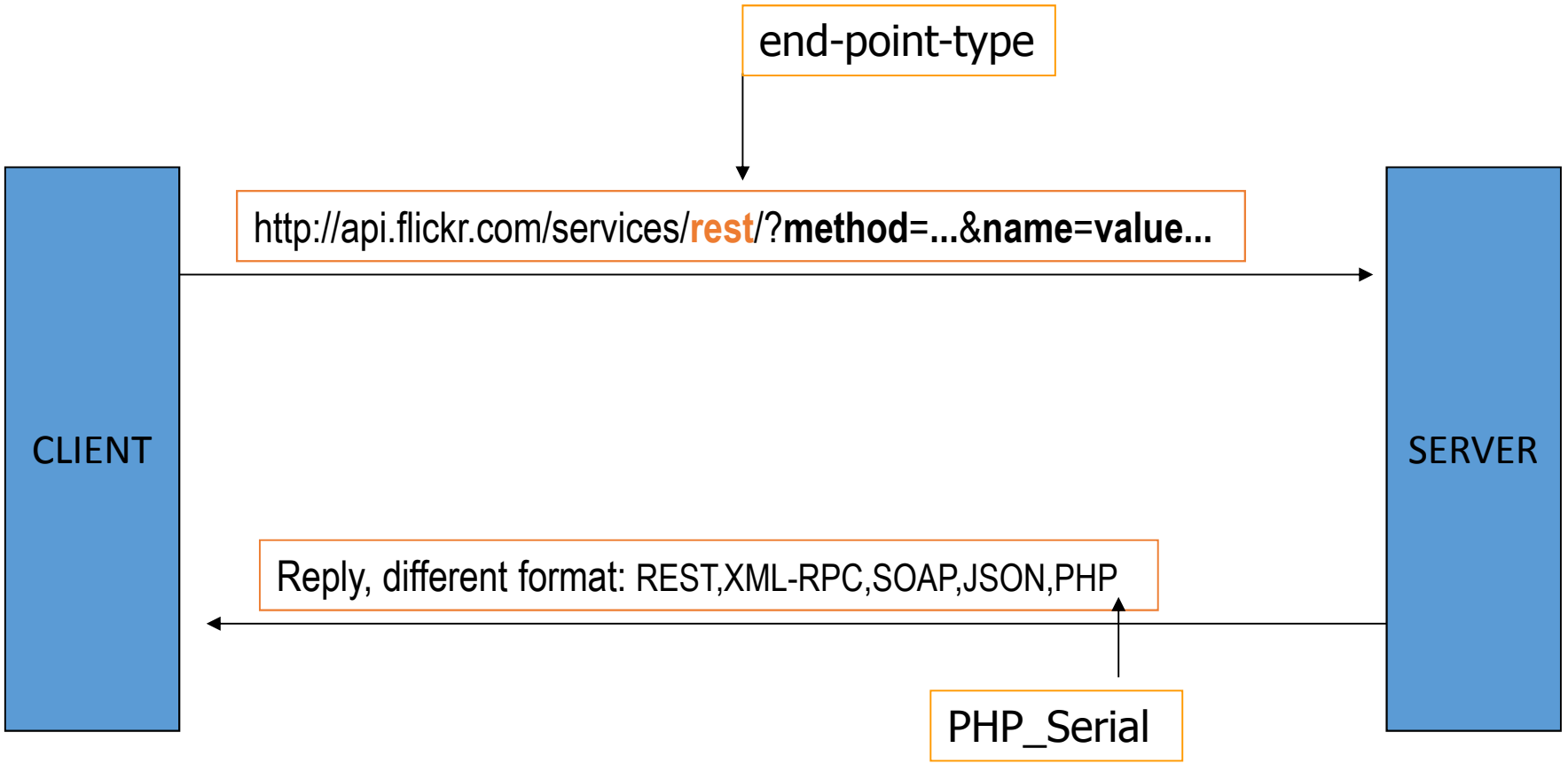
Example

- Flickr is a web site that allows to share personal photos
- Free account for 90 days
- API with different formats
  - Request: REST,XML-RPC,SOAP
  - Reply: REST,XML-RPC,SOAP,JSON,PHP

Flickr’s application end-points

- http://api.flickr.com/services/rest/
- http://api.flickr.com/services/soap/
- http://api.flickr.com/services/xmlrpc/
- http://api.flickr.com/services/upload/
- http://api.flickr.com/services/replace/

REST format is the simplest way; it uses the HTTP POST method



Example of API call

flickr.photos.getInfo

In Parameters:

api_key (Mandatory)
Your API application key.
photo_id (Mandatory)
The id of the photo to get information for.
secret (optional)
The secret for the photo.
If the correct secret is passed then permissions checking is skipped, unless photo is shared.

Out Parameters:

info with different format…

Example of reply

```
- <rsp stat="ok">
  - <photo id="13456789" secret="0b10dfa107" server="11" farm="1" dateuploaded="1115843762" isfavorite="0" license="0" rotation="0" media="photo">
      <owner nsid="78622492@N00" username="channah" realname="Catherine Hannah" location=""/>
      <title>CP</title>
      <description/>
      <visibility ispublic="1" isfriend="0" isfamily="0"/>
      <dates posted="1115843762" taken="2005-05-11 15:36:02" takengranularity="0" lastupdate="1140733254"/>
      <editability cancomment="0" canaddmeta="0"/>
      <usage candownload="0" canblog="0" canprint="0"/>
      <comments>0</comments>
      <notes/>
    - <tags>
        <tag id="460405-13456789-514919" author="78622492@N00" raw="CLH" machine_tag="0">clh</tag>
        <tag id="460405-13456789-4634" author="78622492@N00" raw="Photos" machine_tag="0">photos</tag>
        <tag id="460405-13456789-2620" author="78622492@N00" raw="Spring" machine_tag="0">spring</tag>
        <tag id="460405-13456789-5904" author="78622492@N00" raw="05" machine_tag="0">05</tag>
      </tags>
    - <urls>
        <url type="photopage">http://www.flickr.com/photos/channah/13456789/</url>
      </urls>
    </photo>
  </rsp>
```

An example:
invoking a REST end-point from PHP code

```
$param = array(
            'api_key'   =>      'e568d81ac2ac47e943673641e037be8 c',
            'method'    =>      'flickr.photos.getInfo',
            'photo_id'  =>      '11111',
            'format'    =>      'php_serial',
            );
$encoded_params = array();

foreach ($param as $k => $v)

            $encoded_params [ ] = urlencode($k).'='.urlencode($v);

$url = "http://api.flickr.com/services/rest/?".implode('&',$encoded_params);
```

Parameters

- Reply in php serial format

urlencode

- non-alphanumeric as % sign two hex digits
- spaces as plus (+) signs.

implode

- Join array elements with a string,
- & used as glue string

$url

http://api.flickr.com/services/rest/?api_key=e568d81ac2ac47e943673641e037be8&method=flickr.photos.getInfo&photo_id=11111&format=php_serial

Serialization

string **serialize** (mixed $value)

Generates a storable representation of a value

mixed **unserialize** (string $str)

Creates a PHP value from a stored representation

Invoke method

Transform format into an associative array

```
$ans = file_get_contents($url);

$ans_obj = unserialize($ans);

 if ($ans_obj['stat']=='ok') {

 echo $ans_obj['photo']['id'].'<br>';

 echo $ans_obj['photo']['title'] ['_content'];

 echo $ans_obj['photo']['description']['_content'];

 echo $ans_obj['photo']['dates']['taken'];
}
```