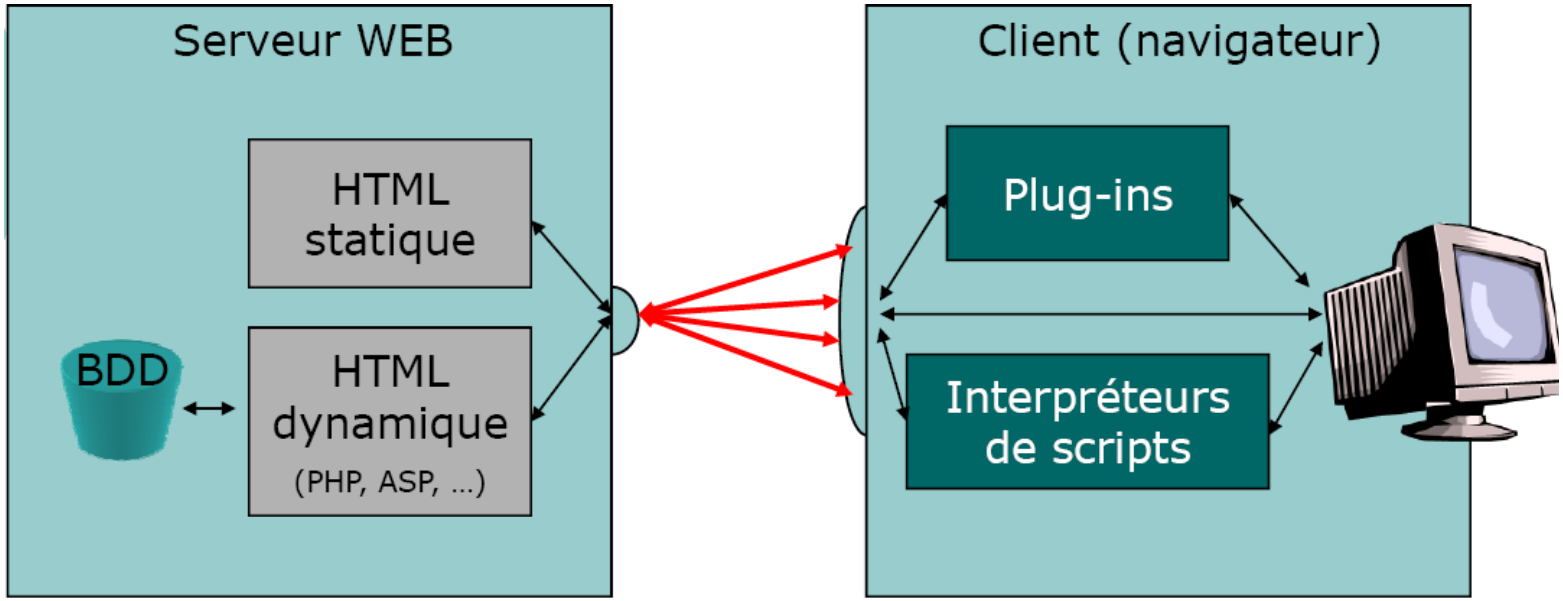
PHP

Dritan Nace

PHP

- Web programming, server side
  - CGI : advantages, drawbacks.
  - PHP : advantages, drawbacks.
- Le PHP : syntax et programming basics
  - Tags, comments, variables, data, instructions;
  - Advanced features, web features;
  - Cookies, sessions, examples.
- PHP : object oriented
  - Review of object oriented programming;
  - Objects in PHP, magic methods, examples.
- PHP review

Web programming, server side



- Server side
  - Static HTML
  - Dynamic HTML
  - Calculation based on elements that the customer has sent to the server in the request .
  - Information from a database, updated by any means.
    - Creation of the HTML code, to be sent to the client

Web Programming

- On the client's side
  - Display, formatting, animation etc.
  - User actions;
  - Data control;
- On the server's side
  - User request (eg. fields in a form) (CGI, PHP, Servlets..)
  - Connexion to a database;
  - Sending of Emails
  - …

CGI

- Common Gateway Interface
  - Principle: a program runs on the server side. It processes specific requests from the user (eg coming from a form) and provides in return html pages to the server, which then returns them to the user.
    - It is therefore an interface / protocol exchange between a form and a server.

- How does it work, technically ?:
  - Via the **get** and **post** methods of a form, requests are sent to the server. The latter locates the CGI and sends it the data as an environment variable
    - For a GET method : QUERY_STRING
    - For a POST method : data is read via the standard input STDIN and the environment variable CONTENT_LENGTH is used.

CGI

- Advantages
  - Use of all languages, therefore the possibility for complex processing;
  - Operational solution in the majority of cases;
  - Compiled language, therefore in general faster than an interpreted language.
- Drawbacks
  - Not very portable, varies from one platform to another;
  - Not very readable, not easily upkept;
  - Consumes system resources : a process is created for each HTTP request.
  - Security concerns/vulnerability:
    - Information can be leaked onto the server;
    - It is possible, via the form fields, to make the server execute commands…
    - When one wants to install a CGI, often the webmasters do not accept…

PHP

- How it works, *technically*
  - PHP works with a module which is added to the web server, downloadable from : http://fr.php.net/.
  - Interpreted language (compiled on the fly);
  - Current version 5.3.2. (4 march 2010)
  - Often used with :
    - Microsoft IIS or Apache HTTPD as an HTTP server;
    - MySQL, PostgreSQL, Oracle or MS SQL as an RDBMS;
    - Windows, linux or Mac OS X for an operating system;
  - Three products : « packages »
    - Easy PHP, WAMP and TSW.

PHP- Operation principle (I)

PHP- Operation principle (II)

- Operation:
  - PHP code is placed directly in the HTML source code, between specific tags;
  - A PHP source file can contain alternating « html » and « php » parts.
  - PHP syntax very similar to C.
  - After processing by the server, the HTML source sent to the navigator (by the server) contains no traces of the PHP code.

PHP : Hello World

- <?php echo 'Hello World !'; ?>

This code allows display on the standard output of the caller.

Integration of PHP code in a web page and generation of HTML code.

- …Hello World !

```
<html>
<head>
<title>Hello World en PHP</title>
</head>
<body>
<p>
<?php echo 'Hello World !'; ?>
</p>
</body>
</html>
```

```
<html>
<head>
<title>Hello World en PHP</title>
</head>
<body>
<?php echo '<p>Hello World !</p>'; ?>
</body>
</html>
```

PHP : syntax and programming basics

- Different means (tags) exist which allow the HTTP to pass control to the PHP module :
  - <?php *insert php code here?> (preferred tags!!)*
  - <script language = "php">
  - *<? insert php code?>*
  - *<% insert php code here%>*

- Comments : « // » ou « /* */ »
- Seperate the PHP instructions by « ; »
- Definition of variables : place a « $ » dollar sign in front;
  - A variable can be used as the name of another variable (with $$)

PHP : data types

- Four simple types:
  - Integers, in php : *int.*
  - Booleans : in php *bool*. They have two values : true and false.
  - Floating point numbers : recognised as real, floating point, or double. PHP type : *float*.
    - Ex. $a = 1.234 (not 1,234).
  - Character chains, in php : *string*.
    - $thing= ‘hello everyone’; or $thing = "hello everyone";
    - A few functions
      - Display with echo, printf, concatenation with « . »…
      - Formatting (strtolower(), strtoupper()); comparison (strcmp()); length strlen(); search in a string strpos(), stripos(); trimming of strings trim()...; substitution str_replace()…

PHP : data types

- Composite types:

  Tables: php type : array. There are numerically indexed arrays and associative arrays (they are practically the same thing !!!) :

  - Declaration and initialisation

```
<?php
$fruits = array();
$legumes = array('carotte','poivron','aubergine','chou');
$identite = array(
'nom' => 'Hamon',
'prenom' => 'Hugo',
'age' => 19,
'estEtudiant' => true
);
?>
```

PHP : data types

- Adding elements:

```
<?php
$legumes[] = 'salade';
$identite['taille'] = 180;
?>
```

- Accessing elements:

```
echo $legumes[2];
echo 'Nom : ', $identite['nom'] ;
```

– Functions related to tables:

  - Counting of elements count(), table browsing reset(), current(), next(), prev(), each(), table sorting asort(), arsort()…uasort()

PHP : data types

- Two special types:
  - Resources
    - PHP type : *resource*
      - Files, databases, connections…
      - File manipulation functions
        - » fopen(), feof(), fgets(), fread(), fwrite(), fputs(),file_exists(), copy(), rename(), opendir(), readdir(), mkdir(), rmdir(),
  - The NULL value , is recognised in php as the null type. It represents the absence of any value.

PHP : operators and instructions

- PHP offers all the classic, standardn, mathematical and logical operators (similar to C).
- Constants : define(constant_name, value)

  ex. define(PI, 3.14159265);

- Conditional instructions
  - ($a > $b) ? $a : $b;
  - If (expression) instruction1 else instruction2;
    - elseif, nested…
    - switch… case case1: instruction… default : instructions.
  - Loops
    - while, do … while, for, foreach,
    - Exiting loops: break, continue, return…

PHP : advanced features

- Defining features :
  - Function($arg1, $arg2, ..)

```
{
$return_value = $arg1.$arg2;
return $return_value ;
}
```

All arguments, except objects are passed by value.

In order to pass them by address, they must be preceded by a &

Visibility of variables in the body of a function :

Three levels : *global*, *static* and local.

Remark. Information concerning functions is available at : www.php.net/nom_function.

PHP : web features

- Generation of dynamic HTML forms (using loops, arrays etc.);
  - Processing data sent by a form.
    - All data sent by a form is automatically available to the specified PHP script in the form, through the superglobal $ _POST [] or $ _GET [] arrays;
    - Preferably POST!!
  - Loading the selected files in a form;
    - An associative array : superglobal ($ _FILES []) allows the retrieval of the files in a form, field (input) type (file), method POST

An example

```
<html><body>
<form method="post" action="verif.php">
Nom : <input type="text" name="nom" size="12"><br>
Prénom : <input type="text" name="prenom" size="12">
<input type="submit" value="OK">
</form></body></html>
```


```
<?php
$prenom = $_POST['prenom'];
$nom = $_POST['nom'];
print("<center>Bonjour $prenom $nom</center>");
?>
```

Bonjour James Bond

PHP : web features

- Sending email
  - Function mail(destinataire(s), subjet, message, optionalHeaders
- Redirecting html pages : use the header function.
  - Ex. <?php header("Location: login.php"); ?>
- Managing cookies
  - Setcookie() function, variable $_COOKIE; all the cookies which are sent to the client are automatically transformed into a php variable.
    - *Careful : cookies should not be used for saving confidential information (passwords for example).*
- Managing sessions
  - Sessions are the equivalent of cookie managing, on the server side

Managing cookies : an example

Creation :

```
<?php
setcookie('designPrefere',‘cielbleu', time()+3600*24*31);
?>
```

Lecture :

```
<html>
<head>
<title>Read a cookie!</title>
</head>
<body>
<p>
Your favorite theme is:
<?php
echo $_COOKIE[‘favoriteDesign'];
?>
</p>
</body>
</html>
```

Erasing:

```
<?php
setcookie(' favoriteDesign ');
unset($_COOKIE[' favoriteDesign ']);
?>
```

Managing sessions (I)

- Managing sessions:
  - A session is a system to keep values from page to page, necessary to carry out a transaction.
  - Sessions are particularly used for these types of applications :
    - Members only areas and secure acces with authentification.
    - Shopping cart managing on an online shopping site.
    - Forms which are spread out over several pages
- Principle: Instead of storing information on the visitor's side (like cookies), they are stored on the server. An identifier is assigned to each client and each time he returns announcing that id, PHP collects all the saved information.

- The identifier of a session may be transmitted either as aCookie (on the client's station) or GET (PHPSESSID added to the end of the URL) or POST depending on the server configuration ...

Managing sessions (II)

- In order to work with sessions, the global variable $_SESSION is needed, functions like :
  - session_start(), session_destroy(), (stop/start a session)
  - session_unset() (destroy all/certain session variables).

- Initialisation (and restoration) of a session
  - session_start()

- Reading and writing of a session:
  - The $_SESSION table : superglobal table associated to the corresponding session
  - Writing in a session : $_SESSION['login'] = ‘dupond';

- Destruction of a session

```
<?php
session_start();
$_SESSION = array();
session_unset();
session_destroy();
?>
```

Sessions versus cookies

- Cookies are stored on the client's side, and sessions on the server's side;
- Cookies are generally less secure than sessions;
  - Cookies circulate the client's info between the server and the client station, whereas sessions indicate that an identifier is valid for a limited time only.
- Cookies are designed to last longer than sessions. If you want sessions that last longer, then databases will be needed to store the client info on the server
- Client side : cookies are stored on the hard drive and sessions in RAM.
- Client side : the session information is stored in temporary files.
- Sessions often need cookies in order to work properly, but it is impossible to do without them…

PHP : advanced features

- Error management
  - PHP can display several error levels (E_ERROR, E_WARNING …)
  - Function : error_reporting();
- Exception management
  - try { … } catch (){…};

Object oriented programming: review

- Object approach basics
  - Abstraction
    - Extract the properties and important methods associated with an object from the real world;
  - Encapsulation
    - Group methods into two broad groups (nucleus et interface)
  - Modularity
    - Decompose a complex system into simpler sub-systems
  - Hierarchisation
    - Classing and ordering these abstractions

Object oriented programming: review

- Five fundamental object concepts:
  - The object
    - An object is the result of abstraction : a software package which groups variables (properties) and functions (methods) which are its own
  - The class
    - The class consists of properties and methods which will be shared by all its object. Has a constructor / destructor.
  - Generalisation and/or specialisation (inheritance)
    - Association of a class with a more general class, called superclasst. Specialisation is the opposite ...
  - Polymorphism
    - Idea : if inheritance concerns classes, polymorphism is related to methods of objects.
  - Messages
    - Ability to allow objects to communicate via messages

Object oriented programming in PHP : classes

- Defining a class in PHP 5 :

```
class voiture{

 public $type = "mercedes";
  public function afficheType () {echo $this->type}
  }
```

  - Instanciate a class

```
$mavoiture = new voiture();
$mavoiture->afficheType();  //display mercedes
```

  - Default constructors/destructors or the functions *__construct() et __destruct();*
  - In order to access attributes (declare them private), getter and setter functions are preferable*;*

- Class visibility:
  - Keywords : *public, protected, private* offer a decreasing visibility level.

Object oriented PHP: an exemple

```
<?php
 class Etudiant {
  public $nom;
  private $semestre;
  private $nbuvchoisi;

  function __construct($nom, $semestre, $nbuvchoisi) {
    $this->nom = $nom;
    $this->semestre = $semestre;
    $this->nbuvchoisi = $nbuvchoisi;
    }

  function arreter() {
    echo "Etudiant $this->nom stopped this course this semester.<br/>";
    }

  function suivisemestre($nb_uv) {
    $this->nbuvchoisi += $nb_uv;
     }

  function etat() {
    echo "Etudiant $this->nom chose to take $this->nbuvchoisi course
       en $this->semestre.<br/>";
    }
  }

$bond = new Personne('Bond', 'GI01', 2);
$bond->etat();
$bond->suivisemestre(2);
$bond->arreter();
$bond->etat();
? >
```

Object oriented programming in PHP

- Inheritance
  - No multiple inheritance;
  - Use the keyword : *extends : class daughter-class extends mother-class*
  - Methods/attributes of the mother-class are available through *parent::*
  - The method calls and member accesses can be overloaded via the __call, __get and __set methods. These methods will only be triggered if the object, inherited or not, does not contain the member or method which we wish to access.

An example of inheritance

```
class myclass {
    var $myvar;
    function myclass($myparameter) {
      $this->myvar = $myparameter;
    }
    function afficher() {
      echo "myvar vaut : $this->myvar";
    }
  }
```

```
<?php
  class myclass2 extends myclass {
    var $myvar2 ;
    function myclass2($myparameter, $myparameter2) {
      $this->myvar = $myparameter;
      $this->myvar2 = $myparameter2;
    }
    function afficher2($color) {
      echo "<font color=\"$color\">myvar vaut : $this->myvar</font><br>";
      echo "<font color=\"$color\">myvar2 vaut : $this->myvar2</font><br>";
    }
  }
?>
```

An example of inheritance

```
<?php
    class addition_class
    {
        function addition_class($a,$b)
        {
            global $somme;
            $somme = $a +  $b;
            return $somme ;
        }

    }

    class moyenne_class extends addition_class
    {
        function moyenne($a,$b)
        {
            global $somme, $maMoyenne ;
            $this -> addition_class($a,$b);
            $maMoyenne = $somme/2 ;
            return $maMoyenne;
        }
    }

    $chiffre_1 = 12 ;
    $chiffre_2 = 15 ;
    $maMoyenne = new moyenne_class($chiffre_1,$chiffre_2) ;
    echo "The sum of the two numbers : $chiffre_1 et $chiffre_2 is $somme <br/>";
    $maMoyenne -> moyenne($chiffre_1,$chiffre_2);
    echo " The sum of the two numbers :  $chiffre_1 et $chiffre_2 is  $maMoyenne ";

?>
```

Object oriented programming in PHP : Polymorphism

- Use abstract classes
  - Declare an abstract class (un-instantiable) (*abstract* keyword)
  - Define classes by inheritance, redefine its functions…
- Use interfaces
  - An interface determines the services which certain classes will be able to carry out.
  - Defined by the keyword interface. The functions are public.
  - A class which implements the methods of an interface is defined by : *class name-class implements name-interface*

Special feature of PHP : magic functions

- The functions __*construct*, __*destruct*,__*call*, __*get*, __*set*, __*isset*, __*unset*, __*sleep*, __*wakeup*, __*toString*, __*invoke*, __ *clone*… are recognised as magic functions in PHP.

- *Magic* methods are called automatically when certain actions are carried out;

Magic functions__set() and __get() : example

```
<?php
class MaClasse{
    private $vars = array();
    public function __construct(){
    }
    public function __set($var, $val){
        $this->vars[$var] = $val;
    }
    public function __get($var){
        if(isset($this->vars[$var])){
            return $this->vars[$var];
        } else {
            throw new Exception("attributs ‘$var’ does not exist");}
       }
    }
$maClasse = new MaClasse();
$maClasse->hello = “world”; // adds the attribute "hello" and initialises it with the value
          “world”.
echo $maClasse->hello; // displays “world”
echo $maClasse->resto; // throws an exception
?>
```

Interfacing with databases(I)

- PHP allows very simple interfacing with 3 steps :
  - Declare the variables which will enable the connection to the database:
    - $user : User name
    - $passwd : Password
    - $host : The host (computer on which the DBMS is installed)
    - $bdd : The name of the database
  - Use functions to manipulate databases (mysql etc ...).
    - The function for the connection to the server (*mysql_connect..).* Processes errors :
      - mysql_connect($host,$user,$passwd) or die ("connection error to $host");
    - The function for the choice of the database (*mysql_select_db, …)*
    - The request function(*mysql_query, …)*
    - The deconnection function (*mysql_close, …)*

Interfacing with databases(II)

- Process errors and collects the results (mysql).
  - Ex. *mysql_connect($host,$user,$passwd) or die(« connection error to the server $host");*
  - Store all the records in a variable array, which will be usable with the function : *mysql_fetch_row()*:

$result = mysql_query($query);

while($row = mysql_fetch_row($result))

{…

… }

And : mysql_close();

New : with PHP 5.1 the extension PDO (PHP Data Object) which constitutes the common basis for acces to DBMSs.

Three classes allow this :

- *the PDO class designed for the connection with the bases;*
- *the PDO class Statement manages requests and the result;*
- *the PDO class Exception allows the processing of errors.*

PHP review

1. PHP : runs on the server, requires the installation of a module (PHP interpreter)
2. Possibility of operations on files.
3. Cookie managment
4. Connection to databases
5. Possibility of creation and managment of images.
6. A language which is well-equipped with functions. More than 3000 native functions available.
7. A language which is evolving.

Bibliography

- PHP & MySQL, L. Isolda et S. Magne, eds. MicroApplication, 2005.
- Les cahiers du programmeur, Zend framework, bien développer en PHP. J. Pauli G. Poncon, ed. eyrolles, 2009.
- PHP 5 avancé, 5eme edition, E. Daspet, C.P. de Geyer, ed. Eyrolles 2009.
- For more information, also consult the Internet sources :

  http://www.developpez.com
  http://fr.wikipedia.org
  http://www.commentcamarche.net/
  http://www.apprendre-php.com/tutoriels
  http://www.phpfrance.com/tutoriaux
  …

An example (www.apprendre-php.com/tutoriels)

Practical case: identification form for a member area

```
<?php
// Definition of the constants and variables
define('LOGIN','toto');
define('PASSWORD','tata');
$errorMessage = '';
// Test the sending of the form
if(!empty($_POST))
{
// Is the identification transmitted?
if(!empty($_POST['login']) &&
      !empty($_POST['password']))
{
// Are they the same as the constants?
if($_POST['login'] !== LOGIN)
{
$errorMessage = ‘Bad login !';
}
elseif($_POST['password'] !== PASSWORD)
{
$errorMessage = ‘Bad password !';
}
else
{
// Open a session
session_start();
// Save the login and session
$_SESSION['login'] = LOGIN;
// Redirection to the file : admin.php
header('Location:
http://www.monsite.com/admin.php');
exit();
}
}
else
{
$errorMessage = ‘Please give your
identification!';
}
}
?>
```

```
<head>
<title>Authentification form</title>
</head>
<body>
<form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>"
      method="post">
<fieldset>
<legend>Identify yourself</legend>
<?php
// Any errors?
if(!empty($errorMessage))
{
echo '<p>', htmlspecialchars($errorMessage) ,'</p>';
}
?>
<p><label for="login">Login :</label>
<input type="text" name="login" id="login" value="" />
</p>
<p>
<label for="password">Password :</label>
<input type="password" name="password" id="password" value="" />
<input type="submit" name="submit" value="Logon" />
</p>
</fieldset>
</form>
</body>
</html>
```

```
<?php
// Continue the session
session_start();
// Test to see if the session variable exists and contains a value
if(empty($_SESSION['login']))
{
// If inexistant or null, redirection to the login form
header('Location: http://www.monsite.com/authentification.php');
exit();
}
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.1//EN"
      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
<title>Administration</title>
</head>
<body>
<?php
// We are logged on, and display a message
echo ‘Welcome ', $_SESSION['login'];
?></body>
</html>
```

PHP : design patterns

- In software engineering, a **design pattern** is a concept designed to solve recurring problems using the object paradigm.
- The Singleton makes it possible to have one and only one instance of a given class in a program.
  - Ex. managing the connection to the database server…

- A singleton contains three characteristics :
  - A private and static attribute which holds the unique instance of a class.
  - A private constructor in order to stop object creation outside of the class
  - A static method which either allows the instanciation of a class or to return the unique instance created.
  - *To see a practical example : http://www.apprendre-php.com/tutoriels/tutoriel-45-singleton-instance-unique-d-une-classe.html*