# Introduction aux applications web

Développement web HENALLUX — IG3 — 2016-2017

# Au programme...

- Du site web statique au site web dynamique
  - Sites web statique, HTML-dynamique, dynamique
- PHP, un premier aperçu
  - À quoi sert PHP ? Comment cela fonctionne-t-il ?
- Architectures distribuées
  - Un peu de vocabulaire pour parler des applications comportent un aspect « client » et un aspect « serveur ».
- Organisation du cours
  - Déroulement du cours, évaluation, etc.

#### Un site web

Comment fonctionne un site web (version IG1) ?

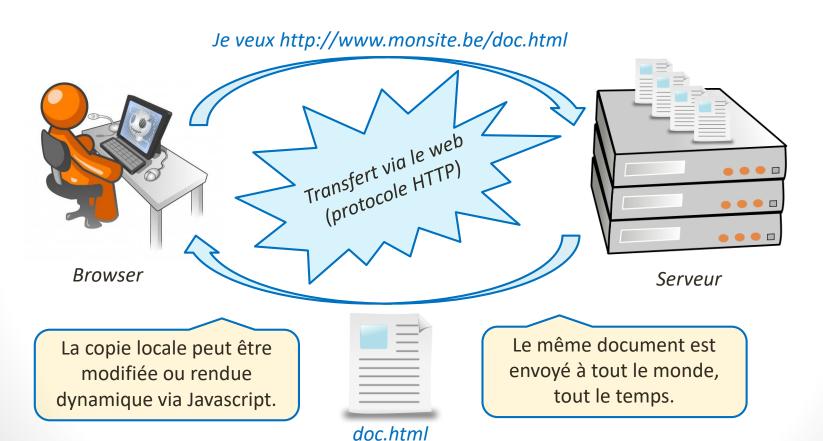


#### Un site web

- Quelles sont les ressources utilisées ?
  - un navigateur utilisé par l'internaute
    - qui envoie les demandes,
    - réceptionne les réponses et
    - gère l'affichage (standards HTML et CSS).
  - un serveur
    - serveur = à la fois machine physique et logiciel
    - machine physique accessible via internet
    - logiciel-serveur capable de répondre aux demandes de fichiers

# Un site web HTML-dynamique

Comment fonctionne un site web (version IG2) ?

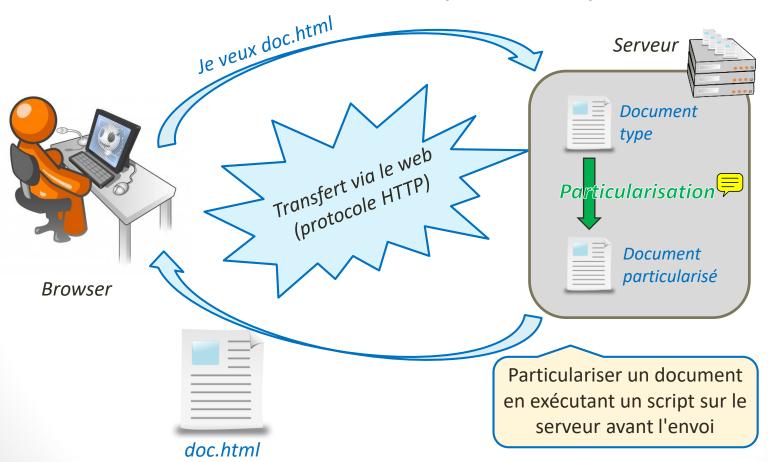


# Un site web HTML-dynamique

- Quelles sont les ressources utilisées ?
  - un navigateur utilisé par l'internaute
    - qui envoie les demandes,
    - réceptionne les réponses,
    - gère l'affichage (standards HTML et CSS) et
    - exécute des scripts Javascript.
  - un serveur
    - serveur = à la fois machine physique et logiciel
    - machine physique accessible via internet
    - logiciel-serveur capable de répondre aux demandes de fichiers

# Un site dynamique

Comment fonctionne un site web (version IG3) ?



# Un site dynamique

- Quelles sont les ressources utilisées ?
  - un navigateur utilisé par l'internaute
    - qui envoie les demandes,
    - réceptionne les réponses,
    - gère l'affichage (standards HTML et CSS) et
    - exécute des scripts Javascript.
  - un serveur
    - serveur = à la fois machine physique et logiciel
    - machine physique accessible via internet
    - logiciel-serveur capable de répondre aux demandes de fichiers
    - logiciel capable d'exécuter des scripts particularisant ces fichiers. C'est là qu'intervient PHP...

# PHP, un premier aperçu

- PHP « in use »
   Un exemple d'utilisation
- Fonctionnement général
   À quoi sert PHP ?
- Utilisations de PHP
   Que peut-on faire avec PHP ?
- PHP: logiciels compagnons et alternatives

Ensuite : Éléments d'architecture distribuée

#### PHP « in use »

Structure du document de base :

```
<h1>Bienvenue !</h1>
Il est actuellement
<?php
  $heure = date('H:i');
  echo $heure;
?>
et tout va bien.
```

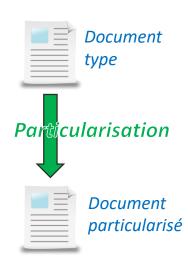
doc.html doc.php

Les scripts PHP peuvent être inclus directement dans le document-type.

#### PHP « in use »

#### • Idée de base :

- Le document de départ sera à la base un document HTML.
- On va rendre certaines parties de ce document "variables".
- Pour définir ces parties, on va donner un script permettant de générer le contenu attendu.



#### • Exemple :

<h1>Bienvenue !</h1>
Il est actuellement 13:44 et tout va bien.

Partie variable générée par un script

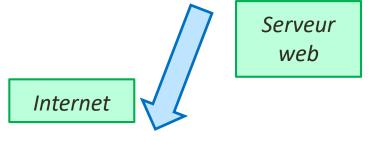
Demander l'heure courante au système. Sortir cette heure au format hh:mm.

#### PHP « in use »

```
Processeur
PHP
```

```
<html>
 <head></head>
  <body>
    <h1>Bienvenue !</h1>
    Il est actuellement
    <?php
     $heure = date('H:i');
      echo $heure;
    ?>
   et tout va bien.
</body>
</html>
```

```
<html>
    <head></head>
    <body>
        <h1>Bienvenue !</h1>
        Il est actuellement
        11:18
        et tout va bien.
    </body>
</html>
```



#### Bienvenue!

Il est actuellement 11:18 et tout va bien.

# PHP, fonctionnement général

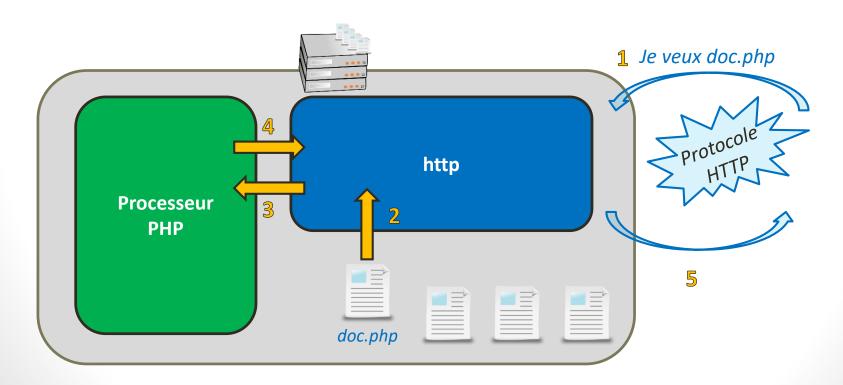
- En cas d'envoi par le client d'une demande pour un document avec l'extension php, ...
  - tout d'abord, le serveur web reçoit la demande ;
  - 2) le serveur web récupère le fichier demandé;
  - 3) il passe la main au processeur PHP capable de comprendre et d'exécuter le code PHP
  - 4) le code PHP est exécuté et remplacé par son résultat ; le fichier résultant (qui devrait être un fichier HTML correct, incluant peut-être du CSS et du Javascript) est renvoyé au serveur web
  - 5) le serveur web envoie le fichier au demandeur via le web.

*Note.* Pour accomplir son travail, le processeur PHP peut éventuellement faire appel à une base de données ou à d'autres éléments présents sur le serveur.

# PHP, fonctionnement général

Comment cela se passe-t-il du côté serveur ?

Note: serveur = machine physique sur laquelle tournent plusieurs programmes qu'on appelle parfois aussi serveurs (http, php...)!



# PHP, fonctionnement général

Deux extrêmes pour le code PHP (selon les goûts) :

```
Tout placer dans le code PHP
```

```
<?php
 echo '<!doctype html>';
 echo '<html><head></head>';
 echo '<body>Je calcule '; Je calcule 17 + 15 =
 echo '17 + 15 = ';
 echo 17 + 15;
 echo '.</body></html>';
?>
```

Balise fermante optionnelle

```
N'y placer que le strict nécessaire
```

```
<!doctype html>
<html><head></head>
<body>
    <?php echo 17 + 15; ?>
  </body>
</html>
```

La balise fermante ?> absorbe le retour à la ligne qui la suit.

PHP permet d'écrire des scripts côté serveur.

Par opposition à Javascript : scripts côté client.

- Le but des scripts PHP est de
  - construire de manière dynamique des pages web,
    - (éventuellement en utilisant une BDD ou d'autres éléments)
  - qui seront transmises au serveur http,
  - qui les enverra aux internautes.

Note. Rien n'empêche un document-type (sur le serveur) de contenir des scripts PHP qui vont générer une page contenant de l'HTML et du Javascript! Et ce code Javascript pourrait émettre des requêtes http qui vont revenir au serveur etc.

- (Jusqu'ici) PHP pour construire des pages web dynamiques
  - Requête : demande d'une page HTML
  - Réponse : un document HTML « particularisé »
- Il peut en faire plus sur le serveur :
  - PHP en tant que langage de programmation à part entière (contrairement à Javascript)
  - Applications plus complexes (gestion de BD) sur le serveur
- Il peut aussi renvoyer autre chose qu'une page HTML :
  - Requête : demande spécifique (évt avec arguments)
  - Réponse : un bout de code HTML

un texte / une chaîne de caractères

un document XML un document json

- Contrairement à Javascript, PHP n'interagit pas directement avec l'utilisateur car PHP agit avant l'envoi de la page.
  - Pas de "alert", pas de "prompt"...
  - Pas d'interaction avec la fenêtre d'affichage, l'historique...
- Contrairement à Javascript, PHP peut
  - créer, ouvrir, lire, écrire et fermer des fichiers sur le serveur ;
  - interagir avec une base de données;
  - contrôler ce à quoi les utilisateurs ont accès (version utilisateur et version admin d'un forum par exemple);
  - •

Note. PHP est également utilisé comme un langage "à part entière", contrairement à Javascript.

- PHP n'a pas de mémoire ("**stateless**") : il oublie tout une fois le boulot terminé!
  - Un navigateur émet une requête pour un document.
    - Le serveur web reçoit la requête et se rend compte que c'est un document php.
    - Il contacte le processeur php qui génère un document html.
    - Le serveur web envoie le document html vers le navigateur.
  - Si le même navigateur ou un autre navigateur émet une nouvelle requête, celle-ci est traitée de manière indépendante de la première : PHP a tout oublié!

# Logiciels compagnons

Que veut dire PHP ?

```
PHP = PHP : a Hypertext Processor
C'est un acronyme récursif, comme GNU (= Gnu is Not Unix) ou
encore Linux (= Linux Is Not UniX).
```

- Logiciel utilisé en conjonction avec PHP : le serveur web
   Apache (« a patchy server »)
  - Plus qu'un simple serveur web
  - Gère automatiquement les fichiers .php (aussi Perl)
- Package all-in-one :
   Wamp = Apache (http) + PHP + MySQL sous Windows (aussi Xampp avec Perl)
   (configuration dans le fichier php.ini)

#### Alternatives

- Autre « monde » du web :
  - ASP.NET (Active Server Page) avec du code C# (généralement)
  - géré par un serveur http IIS (Internet Information Service) ou PWS (Personal Web Server)
- Mais PHP est...
  - relativement simple
  - gratuit et ouvert.
  - portable.
  - très utilisé.

#### Architectures distribuées

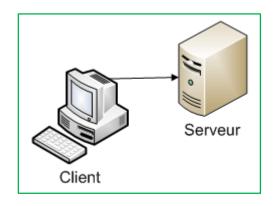
Une application web se divise généralement en deux parties : une partie « client » et une partie « serveur ».

 Quel est le vocabulaire associé à ce genre d'architecture ?

Ensuite: Organisation du cours

# Architecture client/serveur

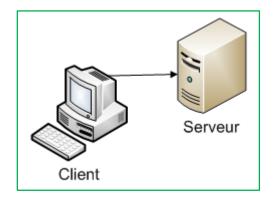
 On qualifie parfois ce genre de structure d'architecture bipartite (ou 2-tiers) ou encore d'architecture client/serveur.



- D'un côté, le client
  - qui initie le processus de connexion au serveur ;
  - qui envoie des requêtes au serveur et traite les réponses.
- De l'autre, le serveur.
  - est à l'écoute des demandes de connexions et les gère ;
  - est à l'écoute des requêtes et les gère.

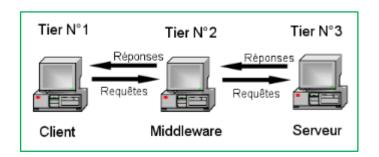
# Architecture client/serveur

Exemples d'architectures bipartites :



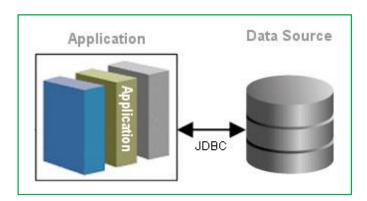
- des navigateurs accédant à un site web situé sur un serveur (version « naïve »)
- des terminaux reliés (physiquement) à un serveur central
- une connexion P2P (peer-to-peer), où les rôles de client et de serveur peuvent alterner

 On ajoute parfois un 3<sup>e</sup> niveau gérant les communications et on parle d'architecture tripartite (ou 3-tiers).



- le client ;
- le middleware, l'intermédiaire entre le client et le serveur.
- le serveur.
- Souvent (mais pas toujours), cette découpe correspond à :
  - client = couche de présentation (« vue »)
  - middleware = couche fonctionnelle de traitement (« contrôleur »)
  - serveur = couche des données (« modèle »)

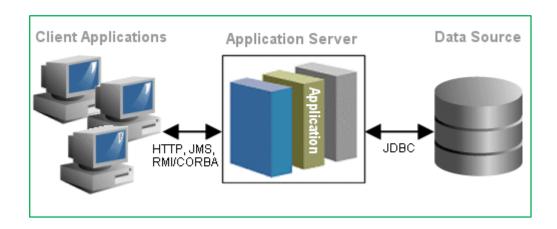
- Exemples d'architectures tripartites :
  - Navigateur ↔ Web ↔ Serveur
     Site web (statique)
     Gadget affichant le cours de la Bourse ou la météo
  - Programme 
     ← Interface 
     ← Base de données locale
     Via Open/Java Database Connectivity (ODBC/JDBC)



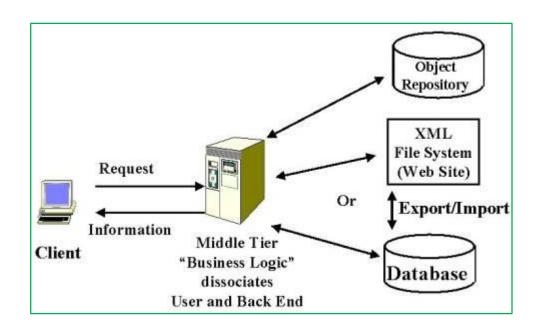
- Exemples d'architectures tripartites (cas courant) :

Site de vente en ligne

Jeu multiplayer en ligne

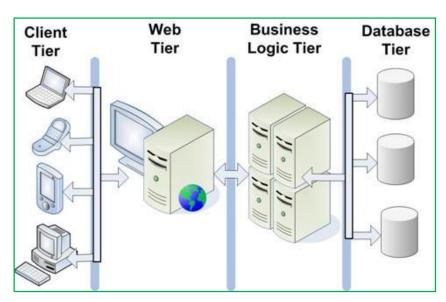


Exemples d'architectures tripartites (suite) :



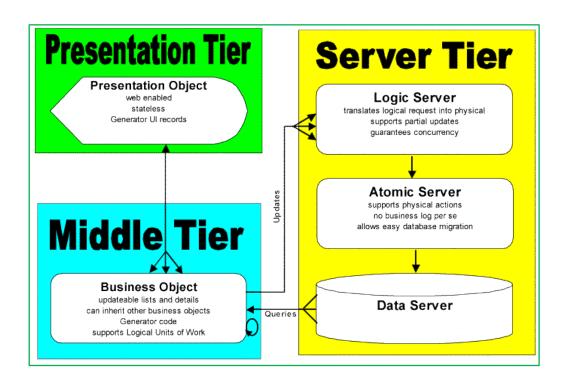
#### Architecture n-tiers

- S'il y a plusieurs autres niveaux intermédiaires, on parle d'architecture multipartite (ou n-tiers).
- Exemples d'architectures n-tiers :



#### Architecture n-tiers

 Un même système peut souvent être découpé de différentes manières selon ce qu'on veut mettre en évidence.



### Un peu plus de vocabulaire...

 Quand une des couches est utilisable/compatible avec plusieurs variantes dans les autres couches, on la qualifie parfois d'universelle.



- Web = middleware universel (grâce à des standards de communication tels que TCP-IP, http, ftp...)
- Site web = serveur universel (grâce à des standards HTML, CSS,
   XML, json... compris par tous [?] les navigateurs)

## Différents types de clients

Le client peut être :

• **léger** : le client se contente de servir d'interface affichant les réponses et envoyant les entrées de l'utilisateur, sans se préoccuper de leur signification.

Tous les traitements sont effectués sur le serveur.

Ex : site web statique

 lourd : le client effectue lui-même une bonne partie des traitements.

Les traitements sont divisés entre le client et le serveur (plus ou moins à parts égales).

Ex : jeux Flash avec high scores conservés en ligne

## Différents types de clients

- Le client peut être :
  - riche: les traitements effectués par le client portent principalement sur l'affichage (graphiques sophistiqués) et la vérification.

La grande majorité des traitements sont effectués sur le serveur.

Ex : widget d'affichage de la météo ou du cours de la bourse

#### Matière du cours

#### Matière

- Le langage PHP
  - Bases du langage
  - Orienté objet en PHP
  - Particularités de PHP liées au web
- Échange de données entre client et serveur
  - GET et POST (révision)
  - Ajax et les requêtes XML via http (HXR)
  - Cookies et sessions
- Inclut également les matières des années précédentes!
  - IG1 (HTML/CSS) et IG2 (Javascript)
  - Munissez-vous de vos notes pour les sessions / l'évaluation !

# Organisation pratique

#### Séances

- Séances mélangeant théorie et laboratoires
  - Introduction
  - Bases du langage PHP
  - Orienté objet en PHP
  - Échanges de données entre le client et le serveur
- Laboratoire
  - Exercices ciblés (temps limité!)
  - Accès à un serveur (logins/mots de passe à distribuer)
- Si possible : conclusion avec un mini-projet...

#### Évaluation

- Examen écrit, à court ouvert, avec PC à disposition
- Matière d'IG1 et d'IG2 incluses !!!
- Attention : cours ouvert ≠ examen pour touriste!