Architecture(s) et application(s) Web

Télécom SudParis

(10/09/2019)

CSC4101 - Cours d’intro, concepts de base, architecture appli Web 3 tiers, PHP

Table des matières

1 Objectifs de cette séance

Cette première séance de cours magistral abordera les éléments suivants :

1. L'organisation proposée pour le cours CSC4101
2. Une présentation du contexte : le Web
3. L'introduction des premiers concepts clés de l'architecture du Web
4. Une présentation de la technologie PHP

2 Objectifs du cours CSC4101

L'objectif de cette section est de présenter la place de ce cours dans l'ensemble des enseignements d'informatique de TSP, et les modalités d'organisation du dispositif pédagogique.

2.1 Enseignement en Informatique (CSC)

Dans le « tronc commun » **Architecture(s) et application(s) Web**
Comment construire une application « classique », sur la plate-forme universelle, le Web, en utilisant une approche maîtrisée.

Ce cours fait suite aux enseignements suivants :

2.1.1 1ère année :

- CSC 3101 Algo. programmation (objet)
  - **algorithmique**
  - **objet**
- CSC 3102 Intro. systèmes d'exploitation
  - **shell**
- CSC 3601 Modélisation, BD et SI
  - **données**, **persistence**, **associations**
- PRO 3600 Projet dév. informatique
  - **interfaces graphiques/Web**

On considérera acquises les notions et savoir faires introduits dans ces modules de première année.

2.1.2 2ème année :

- **CSC 4101 : VOUS ÊTES ICI !**
- …

2.1.3 Objectifs apprentissage de CSC4101

- À l'issue de ce module, les étudiant(e)s […] seront capables de **développer une application Web** de type site d'*e-commerce* (une dizaine de pages), sur la base d'un cahier des charges fourni, en utilisant un *framework* PHP professionnel (Symfony).
  L'application sera réalisée […] en s'inspirant de versions successives d'une application exemple fournie. Elle devra permettre la saisie de données, et aura un comportement personnalisé en fonction du profil d'un utilisateur.
  - ce développement sera effectué par la **mise en œuvre des bibliothèques et outils du *framework* objet**, afin d'exploiter des fonctions de sécurité, de présentation dans des pages HTML, pour s'approcher d'une application réaliste, de qualité professionnelle.
  - ils/elles **utiliseront les outils de mise au point du *framework* et du navigateur** ;
- Les étudiant(e)s sauront **expliquer les rôles respectifs des traitements faits sur le client et le serveur** HTTP.
  - Ils/elles sauront **positionner les composants d'une application Web, dans une architecture en couches (multi-tiers)**.
  - Ils/elles pourront **expliquer comment fonctionnent les sessions applicatives dans un protocole où le serveur est sans-état**.

— Les étudiant(e)s ont la liberté de personnaliser l'apparence des pages du site, ce qui permet **d'appréhender les principes généraux d'ergonomie des interfaces Web** (expérience utilisateur, accessibilité).

3 Organisation de CSC4101

3.1 Planning

3h de cours présentiels / semaine, répartis sur 12 semaines.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
1h	**CM**	CM	CM	CM	CM	CM	CM	CM	CM	CM
2h	TP	TP	TP	TP	TP	TP	TP	TP	TP	TP
3h	*HP*	*HP*	*HP*	*HP*	*HP*	*HP*	*HP*	*HP*	*HP*	

Venez en cours, et en TP !

3.2 10 séquences - semaines

1. Concepts de base, archi. appli 3 tiers, PHP
2. Protocole HTTP, serveur Web
3. Génération de pages, gabarits (*templates*) Twig
4. Expérience utilisateur Web, structure d'une page (DOM), CSS, *BootStrap*
5. Interactions CRUD et sessions
6. Interface dynamique côté navigateur (AJAX)
7. Apports du *framework* Symfony - Autorisations
8. Gestion avancée des routes et formulaires
9. Sécurité, gestion des erreurs
10. Évolution des architectures applicatives

Ceci correspond aux grandes lignes du séquençage du cours. Nous y reviendrons plus tard dans cette séance, plus en détail.

3.3 Hors présentiel

— 10 x 3 h de travail hors présentiel !

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10		
CM TP	CM TP	CM TP	CM TP	CM TP	CM TP	CM TP	CM TP	CM TP	CM TP	…	CF
HP	**HP**	**HP**	**HP**	**HP**	**HP**	**HP**	**HP**	**HP**			

— approfondir les TP (2h de TP, ça va vite)
— **nouvelles connaissances**
— projet (binôme)

Le Hors-Présentiel n'est pas facultatif !

Le temps de travail hors présentiel (HP) fait partie du cours. Il faudra y dédier du temps, dans la semaine, pour des apprentissages en autonomie ou du travail en binôme sur le projet d'application.
Vous êtes incités à utiliser le forum Moodle pour l'entraide et des questions-réponses pendant les phases de travail hors-présentiel.

3.4 Application « fil rouge »

Application Web de **gestion de tâches** : `ToDo`
- fonctionnalités très simples
- PHP + Symfony
- support des TPs
- corrigés au fil de l'eau
- modèle pour le projet

Cette application sera étudiée ensemble en TPs, afin de comprendre les mécanismes de mise en œuvre, et de servir de source d'inspiration pour le projet.

3.5 Projet noté en binômes

Site Web d'Agence d'annonces et réservations de cours séjours, dans des « Couette et café »[1] (C&C) pour les Voyageurs (*aka* `AgVoy`)
Semaines 3 à 10

Un peu comme AirBnB, donc…
Le projet va débuter en séance 3. On reviendra sur ses modalités à ce moment-là.
On peut cependant déjà noter qu'il sera réalisé en binômes et fera l'objet d'une évaluation comptant pour la note du module.

3.6 Outillage

- salles de TP **GNU/Linux**
- *BYOD* (Shell bash, PHP, SQLite, …) ?
  - **GNU/Linux**
  - mac, etc. ?
- *IDE* avec support de PHP, Composer, Twig
  - **Eclipse** (+ PHP Dev. Tools)
  - Atom, VS Code, PHPStorm, … ?
- Navigateur
  - **Firefox**
  - Chrome ?
  - …

1. acception québecoise pour la traduction de *Bead & breakfast* (*B&B*)

Attention : ce module nécessite beaucoup d'activités pratiques en TP ou HP, et l'équipe enseignante n'a pas la possibilité d'effectuer un support sur les machines personnelles en environnements multiples. Nous avons configuré et testé les outils dans un environnement bien défini, mais supporter d'autres configurations avec toutes leurs subtilités n'est pas soutenable pour un module en promo entière.
Nous recommendons fortement l'utilisation d'un système GNU/Linux dans une version équivalente à la configuration standard GNU/Linux proposée par la DISI sur les machines des salles de TP (Fedora).
Même si les technologies PHP étudiées fonctionnent en principe sur d'autres systèmes d'exploitation, nous ne pourrons pas en faire le support.
Il conviendra de vous organiser pour les activités hors-présentielles pour pouvoir utiliser un système GNU/Linux.
Pour celles/ceux travaillant sur des machines personnelles, nous recommendons d'installer un système GNU/Linux dès que possible (en *dual boot* ou machines virtuelles).
De même pour l'environnement de programmation : nous nous appuierons sur les fonctionnalités de l'environnement de développement Eclipse, qui sera équipé des modules *PHP Development Tools* (PDT) et Symfony, afin d'offrir des fonctions de support au développement en PHP, avec Composer, Twig, CSS, YAML, etc.
Vous serez libres d'utiliser un autre environnement, du moment qu'il offre des fonctionnalités équivalentes, mais nous ne pourrons vous assister au niveau de l'équipe enseignantes, si tout ne fonctionne pas comme prévu.

3.7 Évaluation

- Contrôle continu :
  - Rendu projet AgVoy
- Contrôle final

3.8 Équipe pédagogique

3.8.1 Co-coordinateurs

- Christian BAC (C103)
- Olivier BERGER (B303)

Contact :

- `christian.bac@telecom-sudparis.eu`

ET

- `olivier.berger@telecom-sudparis.eu`

Merci de mettre en copie les 2 co-coordinateurs du cours pour toute communication relative à ce cours.

3.8.2 Responsables de groupes

1/A	O. BERGER
2/B	M. GARDIE
3/C	M. SELLAMI
4/D	B. RADDAOUI
6/F	C. BAC
7/G	M. GARDIE
8/H	M. SELLAMI
9/I	E. LALLET

3.9 Comment écrire à son prof

1. Utilisez l'adresse mail `@telecom-sudparis.eu`
2. Donnez un objet clair à votre email
3. Restez simple
4. Utilisez un français correct (ou l'anglais)
5. Soyez agréable
6. Remerciez-les

Just sayin'
Source : Comment envoyer un email à votre professeur ?

3.10 Ressources pédagogiques

- **Présence en cours** :-)
- Moodle et Web site du cours (supports de TP et HP)
- Slides des CM (Web, ou PDF)
- Polys : slides CM + notes rédigées (exemple, pour ce cours)
  - version PDF
  - version modifiable (texte Open Document) pour besoins spécifiques
- **Documentation de référence** (en anglais la plupart du temps)

… et le Web ;-)
Importance de la documentation de référence.
Les polycopiés sont disponible en un seul document (papier / PDF), ou séparément, pour chaque séance en PDF.
Une version au format texte OpenDocument (`.odt`) est également générée pour permettre aux lecteurs de disposer d'une version modifiable, leur permettant d'ajuster finement les paramètres de mise en forme (polices, interlignes, justification, etc.). Celle-ci pourra, nous l'espérons, permettre une adaptation particulière pour des besoins spécifiques (troubles dyslexiques, handicap, …), là où une version PDF standardisée ne conviendrait pas à tous.

4 Concepts clés

On va balayer rapidement les concepts informatiques principaux qui caractérisent le Web.
Ils seront revus et détaillés dans les prochaines séances.

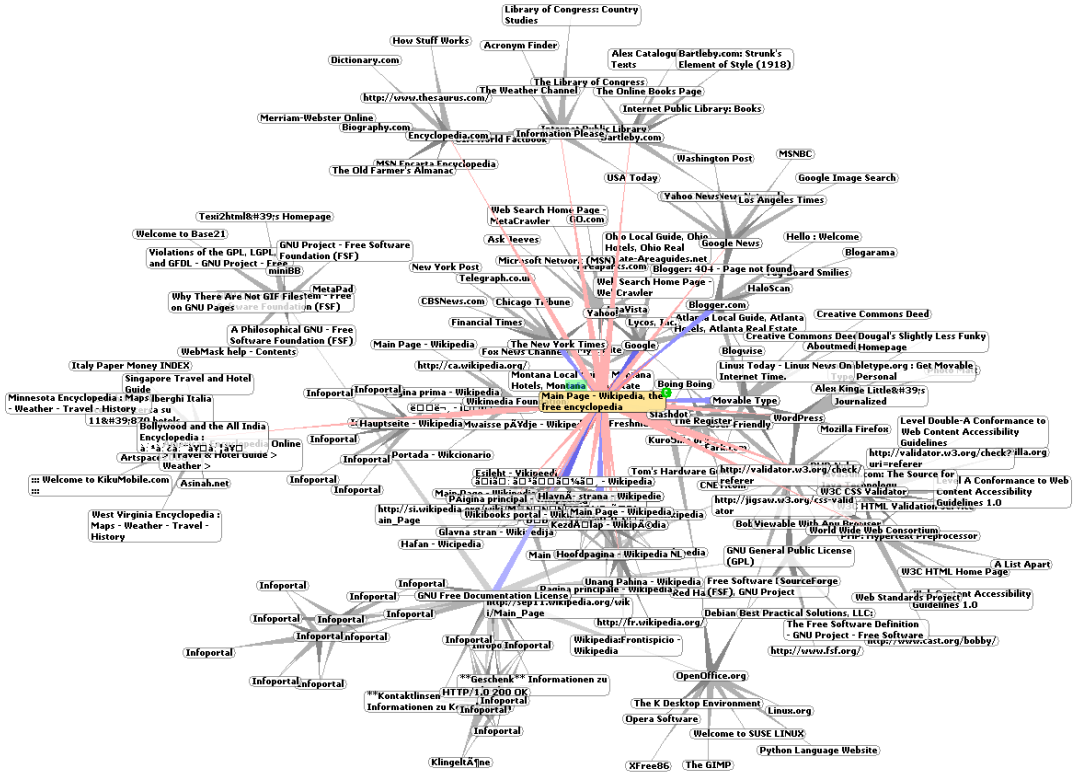
4.1 Le *World Wide Web*

Voyons tout d'abord les principes de conception du World Wide Web initial (avant même qu'il serve à faire tourner les applications)

4.1.1 Histoire

À voir en hors-présentiel (Cf. annexes Poly CM 1)

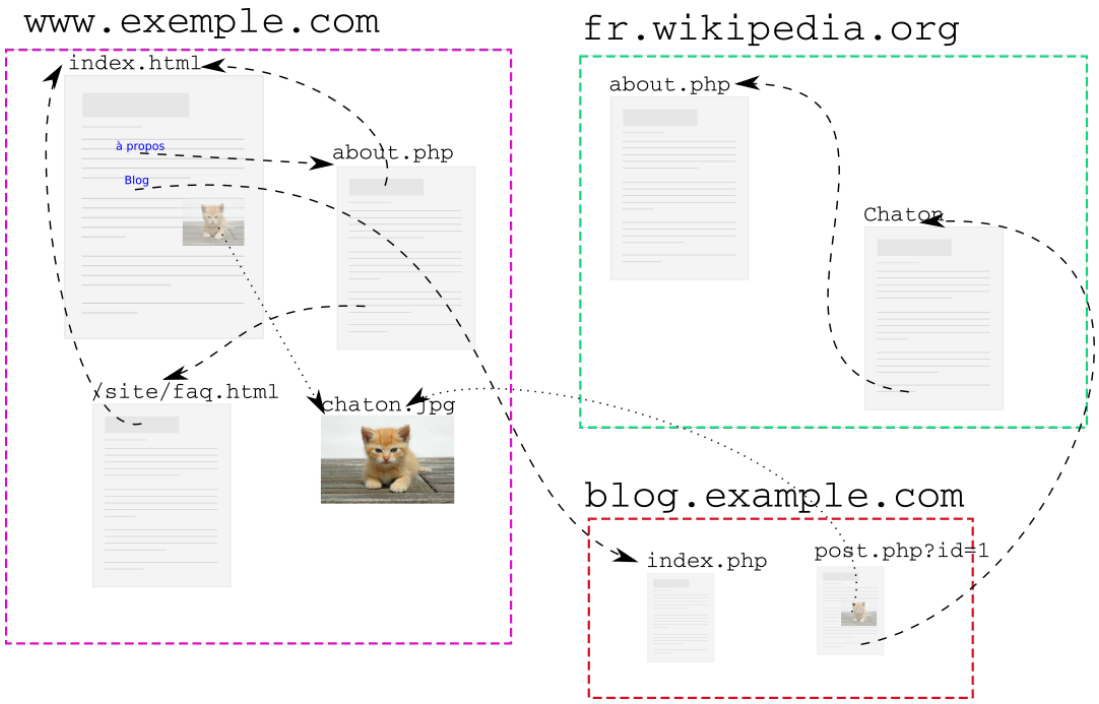
4.2 *Web* = Toile



Source : Chris 73 / Wikimedia Commons

Le *Web* signifie littéralement une toile (d'araignée), en anglais.
Chaque lien pointant d'un site à un autre dessigne une toile mondiale (*world wide web* : `www`).
Ce diagramme illustre le graphe de sites accessibles depus une page de Wikipedia.

4.3 Graphe de ressources liées décentralisé



Le Web est par essence **décentralisé**.
Les liens sont entre différents documents, du même site ou de sites différents.
Certains liens sont navigables, d'autres correspondent à l'inclusion de ressources externes (images).
Note : les liens sont uni-directionnels, du point de vue des serveurs : les documents ne savent pas qui leur pointe dessus. Aucune coordination : pas de permission à demander avant de tisser des liens, mais pas de garantie de disponibilité des ressources distantes.

4.3.1 Ressources

- Documents (statiques)
- Consultation dynamique :
  - ressources issues d'applications (dynamiques)
  - représentation de ces ressources dans les navigateurs Web
- Éléments « virtuels » décrivants des « faits » (Web sémantique/des données)

Pour être précis, on parle de **ressources**, sur le Web. Cette notion est très générique. Elle englobe des documents ou des pages de site, sans exclure des modèles plus avancés d'un Web des données par exemple.
Pour une discussion du terme, d'un point de vue historique, voir par exemple A Short History of « Resource » in web architecture rédigé en 2009 par Tim Berners-Lee.

4.3.2 Ressources liées

- Identification des ressources
- Localisation des ressources
- Agréger des ressources et leurs donner des propriétés pour créer de la connaissance

L'intelligence du Web réside justement dans les liens établis entre les ressources.
Cela permet de présenter des documents riches, mais aussi de naviguer vers d'autres sites.

4.3.3 Décentralisé

- Lier entre-elles des ressources localisées physiquement n'importe où sur Internet
- Confiance, provenance ?

C'est le travail des outils clients, comme le navigateur Web, de construire de l'intelligence en rendant accessible du contenu distribué, mis en ligne de façon pas nécessairement coordonnée.
Rien ne garantit alors que le contenu est fiable, pérenne dans le temps, ou qu'on puisse y faire confiance.
Contrairement au cas d'applications développées dans des environnements contraints et monolithiques, les applications déployées sur le Web doivent ainsi prendre en compte ces contraintes (on y reviendra).

4.4 Clients et serveurs HTTP

Le cœur de l'architecture technique du Web est le protocole HTTP, qui repose sur un modèle client-serveur.

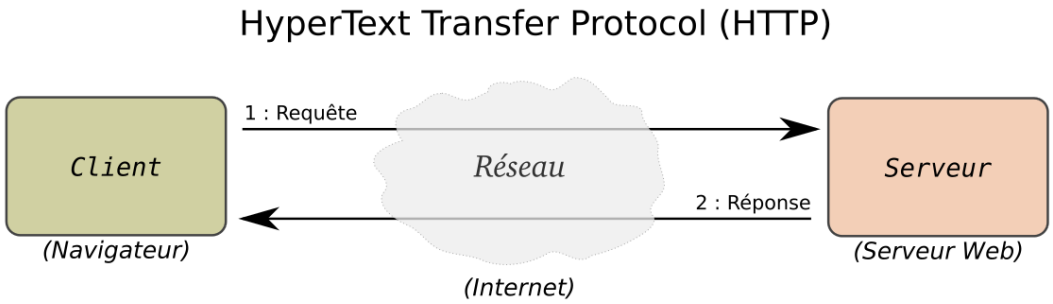
4.4.1 Architecture Client-Serveur

Protocole très simple :

1. requête
2. réponse

Cette architecture est très classique et n'est pas née avec le Web.
Ici, on doit plus précisément parler d'un serveur et de multiples clients qui s'y connectent. Sur le Web, et dans HTTP, les serveurs sont conçus pour communiquer systématiquement avec de multiples clients.
On rappelle qu'il est très fréquent qu'un même client parle avec différents serveurs pour des tâches très courantes, comme la consultation de pages Web dans un navigateur, qui nécessite de charger des contenus sur différents serveurs HTTP indépendants.

4.4.2 Client et serveur Web



En fait : 1 client - n serveurs (modèle distribué)

Le protocole de communication entre clients et serveurs est HTTP (*HyperText Transfer Protocol*).
Le client peut être tout type de programme (navigateur, robot, etc.).
Précision terminologique : dans les spécifications on parle de :
- Client appelé *user agent* dans les spécs
- Serveur appelé *origin server* dans les spécs

La notion d'*origin server* (serveur d'origine), permet de distinguer le serveur d'un éventuel *proxy* (serveurs mandataire) présent entre ce serveur et le client.
Les détails d'HTTP concernant les *proxies* ne seront pas abordés dans ce cours.
On étudiera le protocole HTTP plus en détails dans la prochaine séance de cours magistral.

4.4.3 Dialogue entre client et serveur

- Communication en 3 étapes simples :
  1. Le **client** (navigateur) fait une **requête** d'accès à une **ressource** auprès d'un serveur Web selon le protocole **HTTP**
  2. Le **serveur** vérifie la demande, les autorisations et transmet éventuellement l'information demandée
  3. Le client interprète la **réponse** reçue (et l'affiche)
- On recommence pour des ressource complémentaires (images, …).

4.4.4 Concepts de base d'HTTP

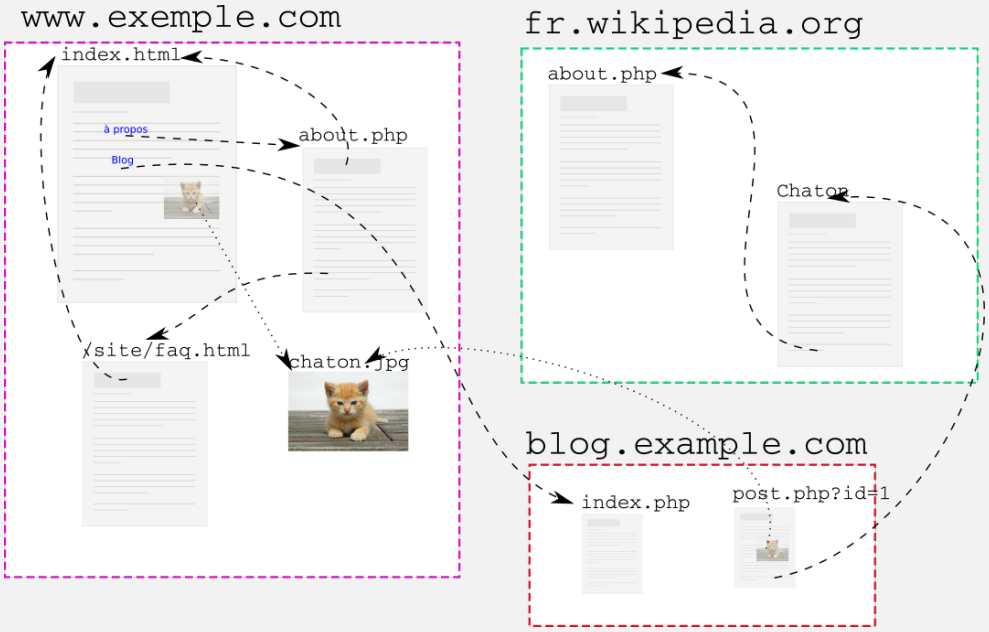
- **Interaction** avec des ressources hypertexte :
  - *Quoi* : **Ressources** accessibles via Internet (documents hypertextes, images, etc.)
  - *Où* : **URL** pour **localiser** les ressources sur des serveurs Internet
  - *Comment* : **HTTP** comme protocole de communication pour un client souhaitant faire des **actions** sur les ressources hébergées sur un serveur

On parle en fait plutôt d'URI que d'URL dans les spécifications, même si pour l'instant cette subtilité a peu d'importance.

4.4.5 Construction de la toile (*Web*)

- **Graphe** de ressources hypertexte/hypermedia **décentralisé**
- Les ressources référencent d'autres ressources (attribut `href` dans HTML)
- Les liens sont unidirectionnels
- Aucune garantie de disponibilité, cohérence
- Parcourir la toile :

1. trouver les liens,
2. accéder aux ressources liées
3. enrichir le modèle applicatif
4. *recommencer* …



Hyper-reference, *hyper-link*
Historiquement, **HTML** comme langage de mise en forme pour la visualisation des ressources de type « document hypertexte »
Le navigateur permet de suivre les liens dans le graphe : le graphe n'existe pas (à part dans un cache) tant qu'on n'a pas essayer de naviguer dessus.
Seul point « stable » : le système DNS, comme pour tout Internet.

- Composants de ce graphe :
  - 3 sites/serveurs Web : `www.example.com`, `blog.example.com`, `fr.wikipedia.org`
    - (peut-être sur 2 serveurs (2 adresses IP) : `example.com` et `fr.wikipedia.org` ?)
  - Documents/ressources :
    - `http://www.example.com/index.html` (HTML)
    - `http://www.example.com/site/faq.html` (HTML)
    - `http://www.example.com/about.php` (HTML)
    - `http://www.example.com/chaton.jpg` (image)
    - `http://blog.example.com/index.php` (HTML)
    - `http://blog.example.com/post.php?id=1` (HTML)
    - `http://fr.wikipedia.org/about.php` (HTML)
    - `http://fr.wikipedia.org/Chaton` (HTML)
  - Liens :
    - Inclusions images
      - Même site
      - Autre site
    - Hyper-liens navigables
      - Document du même site
      - Document sur un autre site
      - Fragment à l'intérieur d'un document

4.4.6 Localisation des ressources

- Objectif : nommer, localiser et accéder à l'information
- Solution : **URL (*Uniform Resource Locator*)** : identification universelle de ressource composée de 3 parties :

1. le **protocole** (*comment*)
2. l'identification du serveur Web, le **nom DNS** (*où*)
3. l'**emplacement de la ressource** sur le serveur (*quoi*)

Plus tard, on raffine avec le concept d'URI : cf. RFC 3986 - Uniform Resource Identifier (URI) : Generic Syntax et standards associées.

4.5 Architecture d'application Web

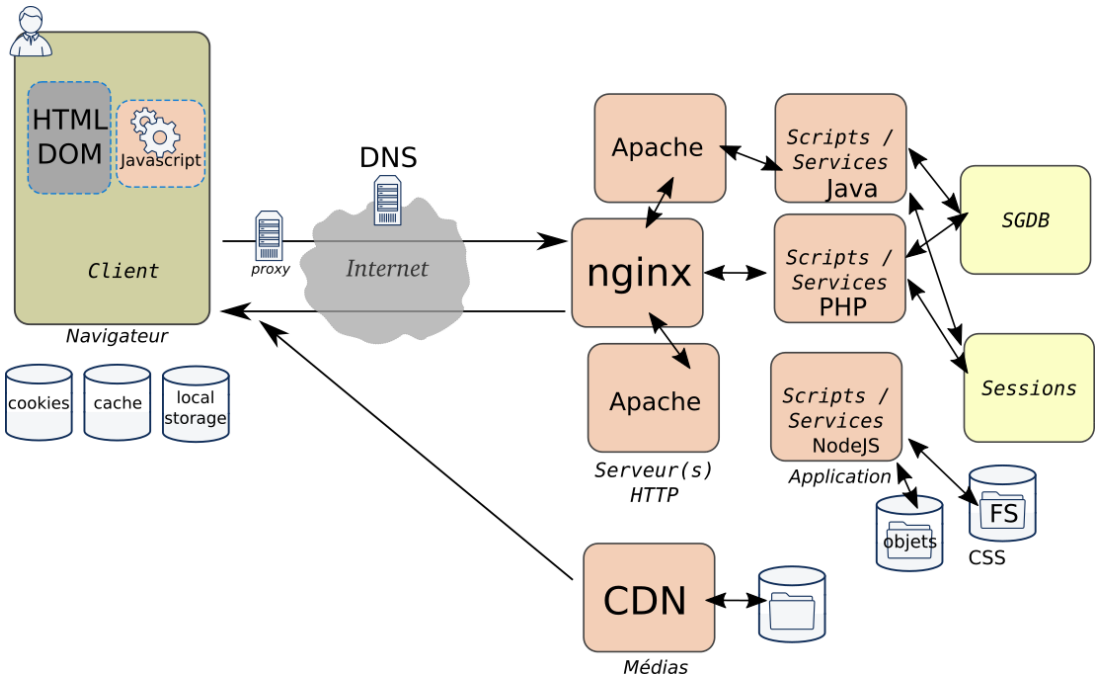
Cette section décrit les grands principes architecturaux qui sous-tendent le fait d'utiliser le Web et HTTP pour faire fonctionner des applications.

4.5.1 Architecture ?

- Structure générale d'un système informatique
  - matériel
  - **logiciel**
  - humain / responsabilités
- Conception
  - Normes ?
  - Standards ?
  - Bonnes pratiques ?

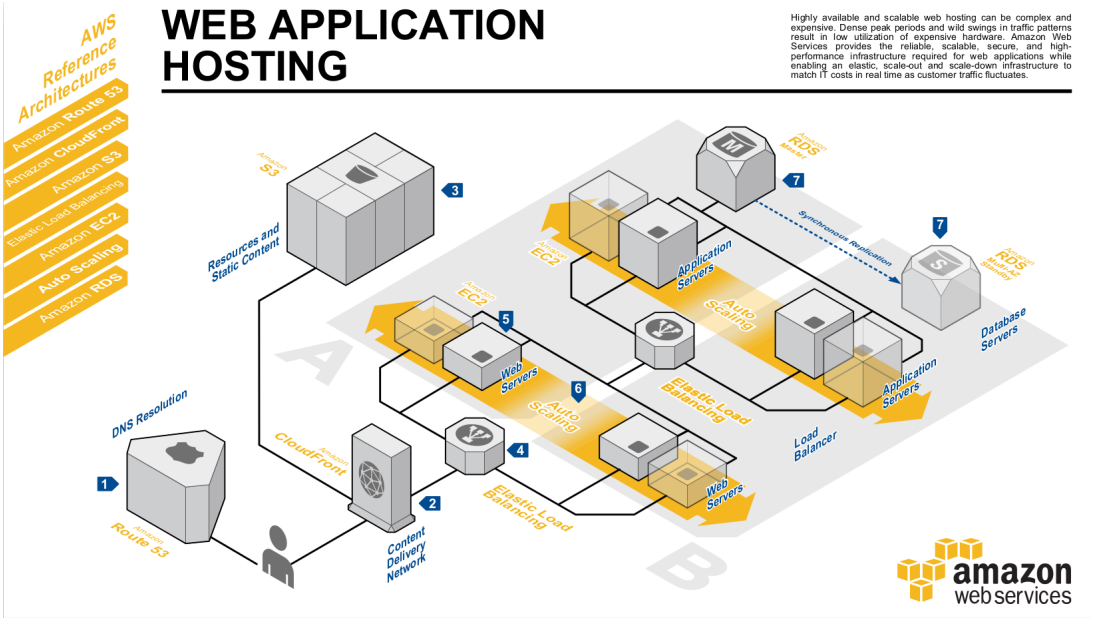
L'intitulé du cours mentionne ce mot « architecture ». Il est donc important de se pencher sur sa signification.
En informatique, comme dans d'autres domaines (bâtiment, …) on peut s'intéresser à des principes d'architecture qui permettent d'apprendre à construire des applications, de la « bonne façon ».
Nous n'étudierons pas en détail toutes les acceptions du terme, mais on verra par exemple, dans la suite du cours, un ensemble de bonnes pratiques consistant à s'appuyer sur un *framework* pour bénéficier d'un ensemble de bonnes pratiques, plutôt que d'avoir à réinventer la roue.

4.5.2 Structure générale



Cette figure illustre de nombreux éléments techniques pouvant intervenir dans le fonctionnement d'une application déployée sur le Web aujourd'hui.
Cette structure est relativement complexe à comprendre, et tous les éléments n'en sont pas essentiel. On a volontairement représenté ici une variante très raffinée, pour montrer la richesse potentielle du domaine.
En pratique, la plupart des éléments présentés sur cette variante ne se justifient que pour des besoins particuliers, par exemple par rapport à des contraintes de performances qui nécessitent un modèle de déploiement spécifique (proxy entrant, …).
On reverra pendant le cours une partie de ces éléments, qui sont essentiels. D'autres éléments seront présentés dans les ressources « pour aller plus loin ».

Plate-forme AWS d'Amazon



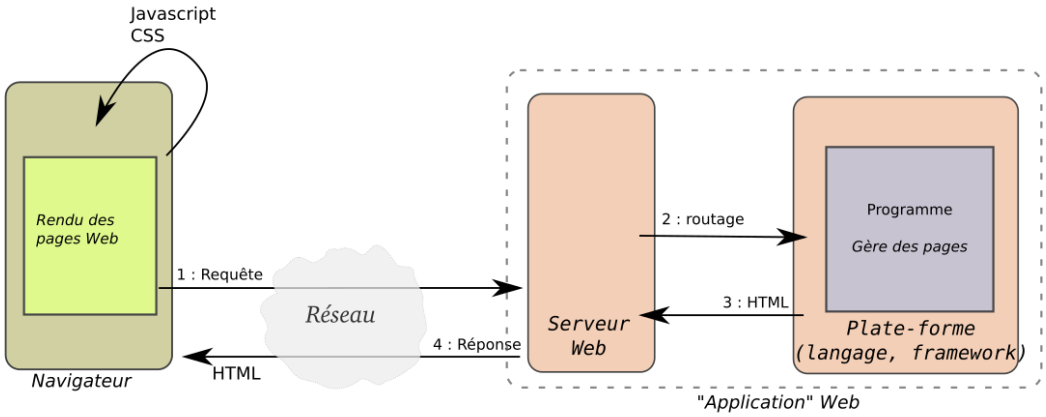
À titre d'information, l'hébergement et la conception des applications Web passe aujourd'hui par des plate-formes dédiées, sur le Cloud, comme celle d'Amazon illustrée ci-dessus.
Cependant, nous n'étudierons pas les propriétés de telles plate-formes dans le cadre de ce cours d'introduction. Nous observerons des plate-formes plutôt plus traditionnelles comme l'hébergement PHP sur un serveur Web « classique ».

4.5.3 Notion d'application

Quésaco ?

Nous avons vu ce que recouvre le mot « architecture »
Interrogez-vous maintenant sur ce que représente la notion d'application (au sens large… ou plus spécifiquement, sur le Web).
Nous détaillerons dans ce cours les grandes propriétés des applications, par exemple dans les propriétés de leurs interfaces pour l'utilisation par les humains (expérience utilisateur).
Nous allons passer en revue très rapidement certaines de ces propriétés, sur lesquels nous reviendrons en détails dans des séances du cours dédiées.

4.5.4 Génération pages HTML à la demande



Ce diagramme illustre l'architecture générale d'une application Web déployée essentiellement côté serveur, qui sera l'architecture de référence que nous étudierons dans ce cours.
Les pages d'une application Web sont construites par une application fonctionnant côté serveur, avec des interactions avec le client navigateur, qui fait aussi fonctionner certains éléments interactifs.
L'essentiel de l'intelligence de l'application, c'est à dire le logiciel qui devra être développé par le programeur, sera déployée dans le programme grisé, du côté de la plate-forme d'exécution d'applications située derrière le serveur HTTP, côté serveur.

4.5.5 Web comme plate-forme

Les technos du Web ne concernent plus seulement les « sites » vus dans un navigateur sur un PC
Importance du côté client :

- **Navigateur**
  - Moteur de rendu HTML
  - Machine virtuelle Javascript
- **Applications natives** programmées HTML5 + CSS + **Javascript** sur mobile
- Client HTTP + (micro) services REST

Native apps vs *Web apps*

L'architecture d'applications présentée dans le cours est très classique et repose sur un modèle un peu ancien, donc éprouvé.
De nouvelles architectures d'applications basées sur les technologies du Web ont vu le jour, mais qui reprennent des concepts similaire. Nous n'aurons pas le temps de les étudier en détails, mais elles reposent sur des variantes des concepts de base que nous aurons étudiés.
Ce que vous aurez appris vous servira (si vous continuez dans le développement d'applications), même avec d'autres architectures apparues plus récemment.

5 Architecture

Cette section présente les principes d'architecture des applications Web classiques, et notamment l'architecture dite 3 tiers.
Les applications Web sont ubiquitaires aujourd'hui, mais on n'a pas forcément toujours des idées très claires sur ce qui permet de faire fonctionner ces applications.
Cette séance va nous permettre d'examiner les mécanismes et protocoles nécessaires au donctionnement d'une application Web.

5.1 Architecture 3 tiers

Architecture *logicielle* :
- couche de **présentation** ;
- couche de **traitement** ;
- couche d'**accès aux données**.

Simplification de l'approche générale d'une architecture à plusieurs niveaux (*multi tier*)

L'objectif de cette décomposition en couches est de mieux comprendre la logique des interactions entre différents composants logiciels mis en œuvre pour faire fonctionner une application. Il s'agit ici d'une architecture logicielle/système, indépendant des fonctionnalités fournies par telle ou telle application (on parlerait d'architecture fonctionnelle).
On parle aussi d'**architecture à trois niveaux** ou **architecture à trois couches**.
Le mot *tiers* est une traduction approximative de l'anglais *tier* signifiant étage ou niveau.

5.1.1 Variante ligne de commande

Application Symfony `MiniAlloCiné` (cf. TP), en ligne de commande :

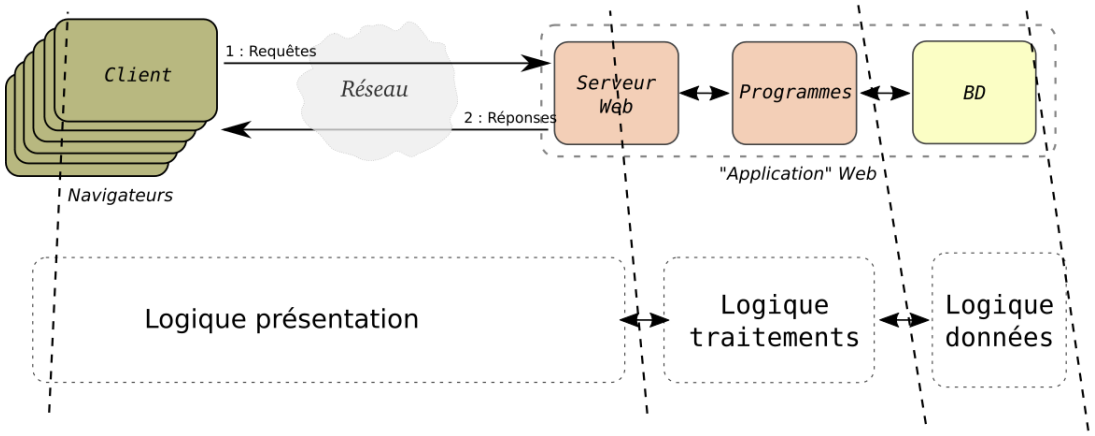
présentation affichage sortie standard

traitement `App\Command\ListFilmsCommand`

accès aux données Doctrine + SQLite (SQL)

Une application démarre pour une exécution pour un utilisateur unique, dans le contexte d'un shell, en ligne de commande dans un terminal.
L'application est un processus de l'interpréteur PHP qui démarre le programme `bin/console` du *framework* Symfony.
L'application est « monolithique », fonctionnant dans la mémoire d'un seul système d'exploitation, en local, en un seul processus.
Les 3 couches sont des couches logiques aidant à comprendre l'architecture fonctionnelle de l'application.

5.1.2 Variante sur le Web



L'application fonctionne sur un serveur accessible depuis le réseau via le protocole HTTP.
Un navigateur Web permet de s'y connecter et de consulter des pages Web présentant le résultat de l'exécution d'un programme fonctionnant dans la mémoire du serveur.
Attention, ici, plusieurs clients Web peuvent accéder simultanément à la même application fonctionnant sur le serveur Web.

5.1.3 3 couches des applications Web

Déploiement sur le Web des différentes couches :

- **Présentation** : *pages HTML* transmises via le réseau :
  - *serveur* qui les construit
  - *client* (navigateur) qui les affiche
- **Traitements** : programmes :
  - serveur Web (dialogue HTTP, invocation des applications)
  - serveur d'applications (exécute des programmes PHP, par exemple)
- **Accès aux données** : SGBD

Les multiples clients Web (de chacun des utilisateurs) communiquent via HTTP avec un serveur Web unique.
Le navigateur Web qui fait le rendu des pages HTML participe à l'application (présentation).
Les clients Web ne sont pas seulement les navigateurs des utilisateurs, mais peuvent aussi être des programmes se connectant à des APIs via HTTP.
Les programmes écrits par les développeurs (par exemple en PHP) fonctionnent sur / derrière ce serveur Web.
Ils peuvent s'exécuter pour effectuer des traitements grâce aux services fournis par des serveurs d'applications sous-jacents (journeaux, envoi de mails, etc.).
Dans cette déclinaison en 3 couches, chaque couche est plus ou moins indépendante des autres. On peut par exemple imaginer que d'autres clients que les clients Web peuvent accéder aux mêmes serveurs d'applications pour interagir avec les programmes (via des protocoles autres que HTTP).
De même, les données stockées dans la base de données peuvent être manipulées par d'autres applications que ces applications Web, ou directement par des utilisateurs via les interfaces du SGBD (cf. CSC3601).

5.2 Évolution des architectures

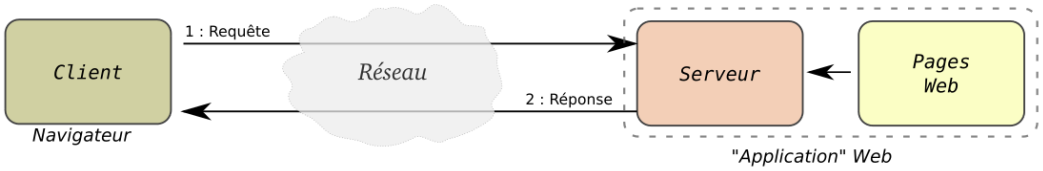
On va présenter les grandes architectures dans un ordre chronologique d'apparition des technologies :

1. pages statiques
2. applications BD + Web
3. multi-couches

La transition du Web statique jusqu'aux applications *multi-tiers* a principalement concerné le côté du serveur Web

5.3 Serveur de pages Web statiques

Historiquement, les premiers serveurs Web se contentent de fournir des pages Web **statiques** chargées depuis des fichiers (documents HTML).



Les fichiers HTML des pages sont stockés sur un système de fichier. Ils sont produits en dehors du périmètre de l'application.

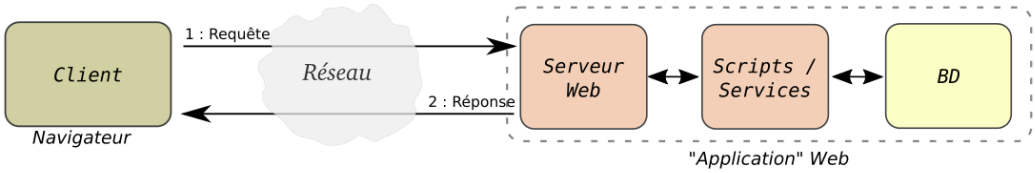
5.3.1 Caractéristiques d'un serveur de pages statiques

1. Avantages
   — Efficace (tient la charge)
2. Inconvénients
   — Ne permet pas facilement des pages qui se mettent à jour dynamiquement, en fonction des visites
   — Encore moins des *applications* interactives
   — Problème de cohérence des URLs (renommages, liens cassés, etc.)

Le problème de cohérence des URLs vient du fait qu'un renommage du nom d'une page, ou son déplacement dans l'arborescence du système de fichiers sur le serveur, nécessite de re-modifier le contenu de tous les fichiers des pages qui pointent vers celle-ci : l'arborescence logique des pages via les liens hypertextes est fortement liée à l'arborescence physique. Des solutions de configuration ou de compilateur de sites permettent de contourner ce problème, mais c'est peu confortable.

5.4 Applications BD + Web

Les serveurs servent à faire tourner des applications produisant des pages Web **dynamiques**, à partir de données issues de bases de données.



BD : Bases de Données. Typiquement via l'interrogation d'un SGBDR
Note *: on ne revient pas sur les technologies de bases de données dans ce cours, que l'on considère comme acquises (programme première année).*
Les programmes qui produisent les pages peuvent aussi utiliser d'autres sources de données que les bases de données

5.4.1 Caractéristiques d'un serveur pour applications BD + Web

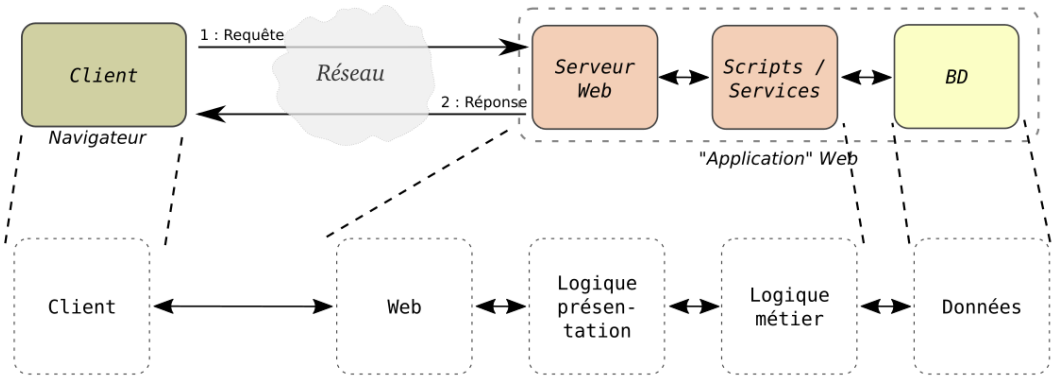
Age d'or du Web : des **applications** sont possibles : des programmes génèrent des pages Web en fonction des requêtes du client

1. Avantages
   - Base de données gère l'intégrité des données (transactions, accès multiples, intégrité référentielle, …)
   - Tient la charge : dupliquer l'exécution des scripts
   - Large gamme de choix du langage de programmation
2. Inconvénients
   - Invocation des scripts par le serveur
   - Difficulté à programmer (+/-)

On ne raffine pas trop le modèle de couches avec l'adaptateur de données qui est parfois introduit, pour ne pas trop complexifier
Les programmes sont appelés *scripts* sur le diagramme, car les langages de scripts (PHP, Perl, Python, etc.) furent très populaire dans l'essor de cette technologie de pages Web dynamiques. Des programmes dans un langage compilé sont aussi possibles (Java par ex.) sans que cela change le modèle.
Différentes techniques d'invocation des programmes par les serveurs Webs (qui étaient initialement dédiés aux pages statiques) ont vu le jour, comme CGI.
Un souci général est les performances pour des consultations simultanées, sachant que tout n'est pas complètement dynamique dans une page Web, mais que si tout est regénéré à chaque consultation, le serveur peut vite s'écrouler.
L'efficacité du mécanisme de déclencement de l'exécution du bon programme, et de récupération du contenu des pages généré est critique. Afin d'optimiser les performances, différentes générations de technologies ont émergé, qu'on verra plus loin.

5.5 Architecture moderne, multi-couches

On distingue les différents composants mis en œuvre dans une application, en différentes couches indépendantes.



Chaque couche s'appuie sur la couche de droite, les flèches illustrant le flux de données
On décrit cette architecture en parlant aussi d'architecture *multi-tier* ou *n-tier*.
Par rapport au schéma de l'architecture 3 tiers précédente, on a distingué notamment la « logique de présentation » de façon indépendante, en la plaçant côté serveur.
Ici, cela correspond à une déclinaison des technologies de construction de pages Web où l'intelligence est principalement gérée par les programmes fonctionnant du côté du serveur, ce qui sera le cas dans les applications que nous étudierons dans ce cours, construites avec le framework Symfony.
Mais on peut trouver d'autres architectures, notamment avec des frameworks Web apparus plus récemment, où le serveur est très peu concerné par la présentation. La logique de présentation s'exécute alors principalement dans des programmes tournant du côté du client Web, en s'appuyant sur les technologies du monde JavaScript.

5.5.1 Caractéristiques

- **Découpage en couches**
- Mieux décomposer l'architecture de l'application côté serveur pour mieux maîtriser le développement
- Découpler par exemple :
  - la **logique métier** (pas nécessairement Web)
  - la **présentation** (sous forme de pages Web)
- Modèle qui peut être raffiné en ajoutant d'autres couches

Cette approche d'architecture multi-couche n'est pas réservée aux applications Web, mais s'applique à d'autres types d'applications.
L'important est la modularisation, le découplage, qui facilite la maintenance, notamment l'évolutivité.
Un autre avantage consiste aussi à permettre de mieux gérer certains problèmes de performances, en scindant les composants d'une couche sur plusieurs exécutions en parallèle, ce qui multiplie les capacités de traitement (pourvu que la cohérence n'en souffre pas, ce qui n'est pas toujours possible).

5.6 Évolutions récentes

Rôle accru du client
- utilisation de JavaScript avec framework (Angular…)
- programmation événementielle sur le client

— page modifiée côté client (DOM)
— micro services côté serveur (API)

Seul l'accès au données (et un peu de logique applicative) reste sur le serveur, et tout le reste dans le navigateur (ou le mobile).

On abordera la programmation côté client en Javascript dans une séance ultérieure.
Ces aspects seront approfondis dans l'UV d'ouverture « Initiation à Javascript et à la programmation d'applications web » (CSC 4531).

6 Technologie PHP

L'objectif de cette section est de présenter un ensemble de technologies de mises en œuvre qui vont nous servir dans les séquences pratiques du cours.
Pour programmer des applications Web, on expérimentera avec le langage de programmation PHP.
Ce cours ne contient pas de manuel d'introduction à la programmation en PHP.
Les bases du langage seront étudiées en travail autonome, sur des ressources externes.
Nous allons présenter ici quelques aspects avancés du langage et de son environnement, non couverts par les tutoriels de base, et qui serviront dans la suite du cours dès la prochaine séance de travaux pratiques.
Il n'est pas utile de comprendre tous les détails dès maintenant, mais de pouvoir s'y référer ultérieurement.

6.1 Faire tourner des programmes sur le serveur Web

- Programmer des applications
- Choix d'un langage
- Choix d'une technologie associée

De nombreux langages et environnement de développement Web sont disponibles.
Nous avons fait le choix du langage PHP et de l'environnement Symfony pour l'application dans ce cours.
Une grande partie des aspects techniques étudiés se retrouveront dans d'autres langages ou frameworks.

6.2 Cadriciel PHP étudié

On parle de *cadriciel* ou « cadre de travail », en anglais *framework*.
Un *framework* fournit beaucoup de fonctionnalités pour diminuer la quantité de code nécessaire à écrire, pour obtenir une application complète.

6.2.1 Symfony 4

"logo Symfony"

- *Framework* de référence
- Assemblage de beaucoup de bibliothèques
- Modèle de composants objet évolué
- Documentation
- Communauté
- Environnement de mise au point

`https://symfony.com/`

On utilisera Symfony 4, la version majeure actuelle, dans le cours.
Attention aux documents ou tutoriaux portant sur des versions antérieures, qui foisonnent sur le Web, et ne s'appliquent pas forcément à cette version récente de Symfony.

6.2.2 Une *licorne* française - SensioLabs

`https://sensiolabs.com/`

SensioLabs "logo Sensiolabs"

« SensioLabs, une des plus belles réussites françaises du web »

GOUVERNEMENT.fr "HistoiresdeFrance, chapitre 21"

#HistoiresdeFrance, chapitre 21

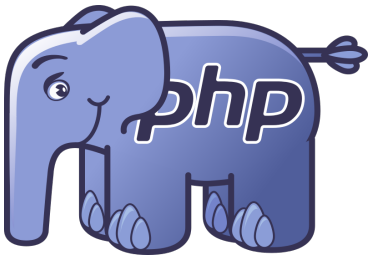
6.3 Autres frameworks :

PHP Laravel, CodeIgniter, CakePHP, Zend, …
Ruby Ruby on Rails, Sinatra, …
Node.js Meteor, Express.js, …
Python Django, Flask, …
Java Spring, Struts,

6.4 Développer en PHP

L'apprentissage des bases du langage sera fait en autonomie dans la prochaine séance hors-présentielle.
On présente ici des informations générales sur le langage.

6.4.1 PHP



"elePHPant"

- Langage interprété
- Logiciel libre
- Version : 7.x
- Bibliothèques
  - Langage
  - « Tierces » (écosystème)

PHP est l'un des langages les plus populaires sur le Web. Cf. Usage statistics and market share of PHP for websites pour quelques éléments chiffrés.
Les versions de PHP que vous allez utiliser devraient typiquement être PHP 7, comme sur machines DISI salle de TP
Pour plus de détails sur les *elePHPants*, voir Comment et pourquoi la mascotte PHP est-elle venue à la naissance ? L'histoire secrète d'ElePHPant ! et A Field Guide to Elephpants - Detailing the attributes, habitats, and variations of the Elephpas hypertextus.

6.4.2 Langage

« *PHP : Hypertext Preprocessor* »

”logo PHP”

- Syntaxe style C / Java
- Objets
- Interprété
- Héritage contexte Web, CGI
- Depuis 1994 (PHP 7 depuis 2016)

6.4.3 *Hello world*

```
<html>
  <head>
    <title>Test PHP</title>
  </head>
  <body>
    <?php echo '<p>Bonjour le monde</p>'; ?>
  </body>
</html>
```

ou bien :

```
<?php

  $html = "<html><head><title>Test PHP</title></head><body>";
  $html .= "<p>Bonjour le monde</p>";
  $html .= "</body></html>";
  print($html);
```

PHP permet d’écrire des morceaux de programme à l’intérieur de pages HTML (*inline*).
Que peut-on en penser ?

6.4.4 PHP *inline*

- Mélanger la présentation (HTML) et le code (PHP)… **c’est mal** : maintenable ?
- On verra comment faire autrement dans une prochaine séance.

Cf. les gabarits (*templates*) en séance 3.

6.4.5 Syntaxe

vous allez travailler dessus en hors-présentiel cette semaine

6.4.6 La documentation

- Le site de PHP : `http://php.net/`
- La documentation : `http://php.net/manual/fr/`

La documentation de référence est à préférer, en cas de doute, notamment pour les fonctions de la bibliothèque standard.
Cependant, on peut douter de sa qualité pédagogique pour l'apprentissage.
On verra que la documentation de Symfony est par contre d'excellente qualité en général.

6.5 PHP moderne

PHP est un vieux langage, et de nombreux tutoriaux ou manuels expliquent comment le prendre en main.
PHP est parfois dénigré car il supporte encore des façons de programmer qui sont déconseillées depuis longtemps.
Ce n'est pas une raison pour jeter PHP à la poubelle. Il y a de bonnes pratiques pour utiliser PHP de façon moderne et produire des programmes de bonne qualité.
Examinons rapidement quelques éléments que nous reverrons plus en détail dans les phases pratiques du cours.

6.5.1 PHP « comme il faut »

S'appuyer sur un *Framework* moderne comme Symfony
- orienté objet
- fonctionnel
- injection de dépendances, conteneurs
- *templates*
- PHPDoc
- tests
- …

`http://www.phptherightway.com/`

La plupart des cours ou tutoriaux qu'on trouve sur PHP sur le Net datent un peu… pourtant il existe d'excellents documents actuels, comme le site ci-dessus « *PHP the right way* » de Josh Lockhart *et al.*

7 Take away

- Structure de la toile
  - Ressources
  - Ressources liées
  - Décentralisé
- URL
- Protocole client-serveur HTTP
- Applications générant des pages Web
- Modèle de couches (3 et +)
  - présentation
  - traitement
  - accès aux données

8 Aller plus loin

Le lecteur intéressé pourra consulter le document Web Architecture from 50,000 feet rédigé par Tim Berners Lee il y a 20 ans. Il date un peu pour certains aspects, mais l'essentiel est toujours applicable, étonamment.

9 Postface

9.1 Slides HTML

On utilise reveal.js pour les *slides* vidéo-projetés en cours.
drinking your own champagne !

Le système de génération des polycopiés et des slides depuis la même source est publié sur : `https://olberger.gitlab.io/org-teaching/README.html`

9.2 Crédits illustrations et vidéos

- Vidéo « Birth of the World Wide Web » © Computer History Museum (used by courtesy of Computer History Museum).
- Illustration plate-forme Web Amazon AWS : `http://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf`
- Illustration « respecte tes parents… » : origine difficile à tracer
- Diagramme « Perdu sur le Web » : `#Geekscottes` par *Nojhan* `https://botsin.space/@geekscottes/101748180337263915`
- Illustration « chaton » : memegenerator.net
- « WorldWideWeb Around Wikipedia – Wikipedia as part of the world wide web » Chris 73 / Wikimedia Commons GFDL 1.3 or CC BY-SA 3.0

10 Annexes

10.1 Structure des URLs

— URL type :

— Composantes :

1. protocole : en majorité `http` ou `https`
2. adresse / nom du serveur : `www.monsite.fr`
3. chemin d'accès à la ressource / document : `/projet/doc.html` (ne garantit pas que le résultat est un document HTML)

10.1.1 URLs plus détaillées

Exemple d'URL plus complexe :

1. protocole : `http` ou `https` (mais aussi `ftp`, `file`, `mailto`, `gopher`, `news`,…)
2. autorité : nom du serveur : `www.monsite.fr` + port : `8000`
3. chemin d'accès à la ressource : `/projet/doc`
4. requête : deux paramètres : `id` et `f`
5. fragment : `num42` à l'intérieur du document

Il existe aussi un sur-ensemble des URLs, les URI (*Uniform Resource Identifier*). On se préoccupe plus d'identifier une ressource que de savoir comment y accéder. En pratique, cette subtilité ne change pas grand chose dans le cadre de ce cours.
Pour plus de détails, voir la spécification générale des URIs (RFC 3986).

10.1.2 URLs absolues ou relatives

— Les URL permettent d'établir des **liens** entre documents
  — sur le même serveur
  — entre différents serveurs
— Liens sur le même serveur :
  — chemin absolu / chemin relatif
  — analogie avec chemin de fichier Unix : `.`, `..`, `/`
  — lien intra-document : fragments/ancres : `doc.html#conclusion`
  — convention : lien dans l'espace d'un utilisateur : `~taconet/menu.html`

10.1.3 Exemples

Document source	Ressource liée	Emplacement réel
`http://w.e.c/index.html`	`about.php`	`http://w.e.c/about.php`
`http://w.e.c/about.php`	`/`	`http://w.e.c/` (contenu de `index.html`)
`http://w.e.c/site/faq.html`	`../index.html`	`http://w.e.c/index.html`
`http://w.e.c/index.html`	`./chaton.jpg`	`http://w.e.c/chaton.jpg`
`http://w.e.c/index.html`	`http://b.e.c/`	`http://b.e.c/` (contenu de `index.php`)
`http://w.e.c/about.php`	`/site/faq.html`	`http://w.e.c/site/faq.html`
`http://b.e.c/post.php?id=1`	`http://w.e.c/chaton.jpg`	`http://w.e.c/chaton.jpg`
`http://b.e.c/post.php?id=1`	`http://f.w.o/Chaton`	`http://f.w.o/Chaton`
`http://f.w.o/Chaton`	`/about.php`	`http://f.w.o/about.php`
`http://f.w.o/Chaton`	`/about.php#terms`	`http://f.w.o/about.php` (`<div id`« terms »>)

10.2 Historique du Web

Cette section présente l'historique du Web, sans entrer dans les détails techniques.

10.2.1 Historique

> 25 ans

1. Avant le Web
   — Minitel (1980-2012)

"Minitel 2"

   — Systèmes *hypertextes* locaux
   — FTP, Usenet, Gopher

   Pour voir des minitels, cf. expo du « Musée de l'INT »
   Gopher voit le jour à peu près au même moment que WWW, et fournit aussi un système hypertexte.
   Il semble avoir subi la concurrence du Web, du fait d'une tentative de gestion de la propriété intellectuelle associée par l'université d'origine, mais aussi plus probablement du fait de l'ajout du support des images dans WWW, alors que Gopher était essentiellement textuel.

2. Hypertexte
   — *Memex* de Vannevar Bush, dans *As We May Think* (*Atlantic Monthly*, 1945)
   — *Xanadu* de Ted Nelson : `https://fr.wikipedia.org/wiki/Projet_Xanadu` (1965-…)
   — *HyperCard* de Bill Atkinson (Apple, 1987)
   — *World Wide Web* de Tim Berners-Lee (CERN, 1989)

   Plus de détails dans `https://fr.wikipedia.org/wiki/Hypertexte`

3. Naissance du World Wide Web
   — Le Web est né au CERN en 1989-1990.
   — Tim Berners-Lee a défini une architecture pour accéder à des documents liés entre eux et situés sur des serveurs reliés par Internet (*Web* = toile d'araignée)
   — le W3C (*World Wide Web Consortium*) mis en place rapidement (1994) pour définir des standards (ouverts).

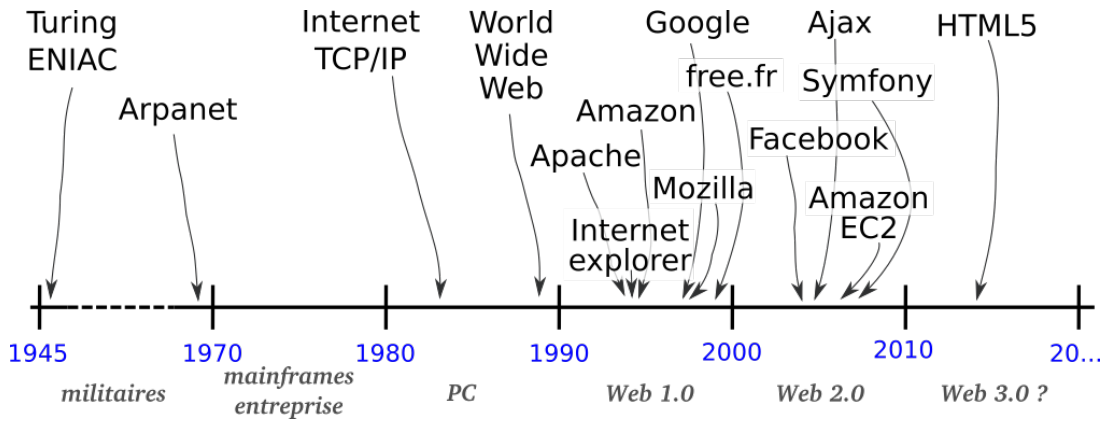
   Objectif : répondre à leur besoin d'échanges de documents (rapports, croquis, photos…) entre des équipes internationales.
   Note : l'IMT est membre du W3C.
   Important : ouverture des protocoles, standards ouverts.

4. Naissance du World Wide Web
   Vidéo Naissance du World Wide Web (extraite de l'expo Web du *Computer History Museum*) `https://www.youtube.com/watch?v=_mNOXDbXr9c`


”Tim Berners-Lee”

5. Premier site
   Il existe toujours, pour les curieux
   `http://info.cern.ch/hypertext/WWW/FAQ/Bootstrap.html`

10.2.2 *Timeline*



— Turing, ENIAC : 1946
— Arpanet : 1969
— TCP/IP : 1983
— WWW : 1989
— Apache : 1995
— Amazon : 1995
— Internet explorer : 1995
— Mozilla : 1998
— Google : 1998
— free.fr : 1999
— Facebook : 2004
— Ajax : 2005
— Amazon EC2 : 2006
— Symfony : 2007
— HTML5 : 2014

Voir aussi `http://webdirections.org/history/#0`

10.2.3 Grandes étapes de l’évolution du Web

1. 1. Naissance du Web

(début des années 1990)
— Accès à des documents structurés via des liens hypertextes
— Protocoles et langages simples
— Technologies de base HTML, HTTP, MIME, formats GIF…

2. 2. Ouverture, homogénéisation et programmation
(fin des années 1990)
— Interactions avec les applications et programmation Web
— Langages plus riches, manipulation d'objets, développement des styles
— Evolution des technologies : XML, CSS, DOM, Server Pages, JavaScript …
— Standardisation difficile (guerre des navigateurs)

3. 3. Evolution des usages et de l'interface utilisateur
(depuis 2005)
— Partage d'informations, édition collaborative, sites communautaires
— Réseaux sociaux, mondes virtuels
— Technologie AJAX, HTML 5
— Intégration de flux RSS, de vidéos, de podcasts
— Personnalisation des accès
— *User-Generated* Content (UGC)

Copyright

Ce cours est la propriété de ses auteurs et de Télécom SudParis.
Cependant, une partie des illustrations incluses est protégée par les droits de ses auteurs, et pas nécessairement librement diffusable.
En conséquence, le contenu du présent polycopié est réservé à l'utilisation pour la formation initiale à Télécom SudParis.
Merci de contacter les auteurs pour tout besoin de réutilisation dans un autre contexte.