Architecture Multi-Niveaux

Patrick FELIX (felix@labri.fr)
Franck RUBI (rubi@labri.fr)
Département Informatique
IUT Bordeaux1

Plan

1. Introduction : vers une architecture n tier
2. Internet, Architecture TCP/IP, Client / Serveur…
3. World Wide Web (WWW) et Protocole HTTP
4. Documents web statiques (XHTML,CSS)
5. Développement web
6. Documents web dynamiques ‘côté client’ & ‘côté serveur’
7. Javascript,DOM,Ajax…
8. PHP
9. XML
10. ASP.NET

Introduction : Vers une architecture multi-niveaux (ou n-tier*)

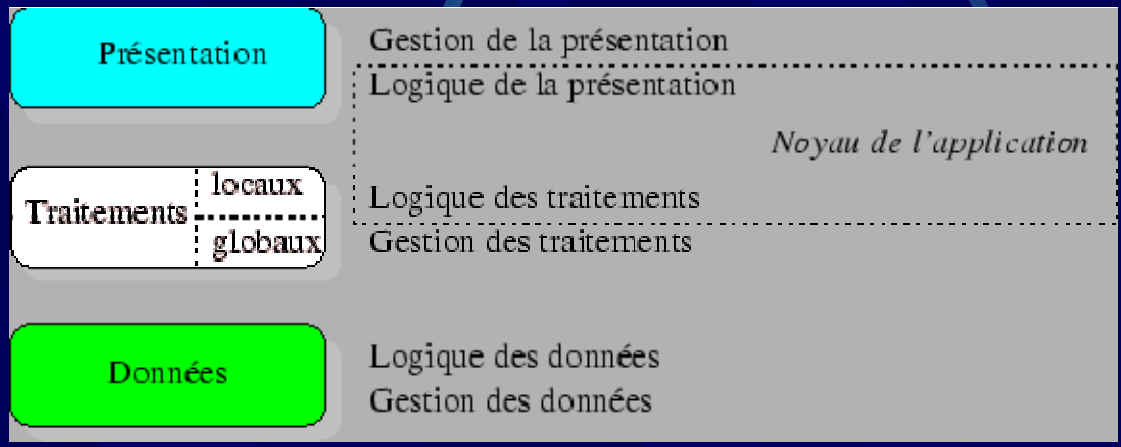
Inspiré de http://remi.leblond.free.fr/probatoire

* Le terme « tier » est abusif, il vient du mot anglais qui veut dire « niveau » et n'a donc rien à voir avec 1/3…

Une application Informatique

Contient en général 3 niveaux différents d'abstraction :

- **La couche de présentation**, ou IHM (Interface Homme-Machine) : interaction avec l'utilisateur.
- **La logique applicative, les traitements** (2 familles) :
  - **les traitements locaux** : contrôles du dialogue avec l'IHM
  - **les traitements globaux** : application elle-même (couche métier)
- **Les données** : accès aux données et leur stockage.

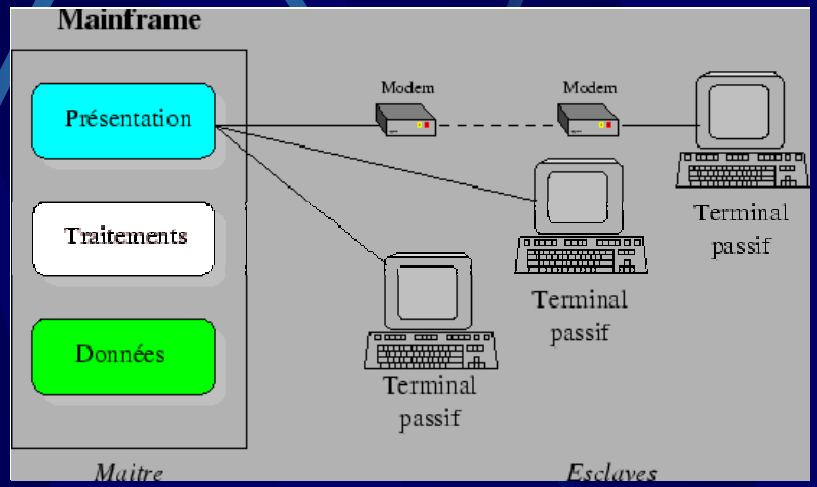


- Ces 3 niveaux peuvent être imbriqués ou répartis de différentes manières entre plusieurs machines physiques.

- Le noyau de l'application est composé de la logique de l'affichage et la logique des traitements. Le découpage et la répartition de ce noyau permettent de distinguer les architectures applicatives suivantes :
  - l'architecture 1-tier
  - l'architecture 2-tier
  - l'architecture 3-tier
  - les architectures n-tier

Architecture 1-tier

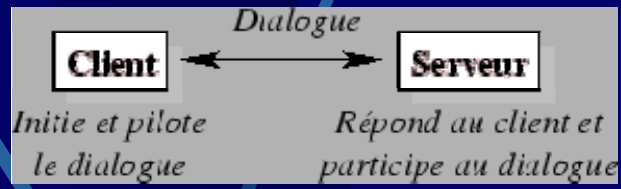
- Les 3 couches applicatives sont intimement liées et s'exécutent sur le même ordinateur. On ne parle pas ici d'architecture Client-Serveur, mais d'**informatique centralisée**.
- Dans un contexte multi-utilisateurs, on peut rencontrer 2 types :
  - des applications sur site central
    - Fiabilité des données mais…
    - IHM (caractères) très limitée…



  - des applications réparties sur des machines indépendantes communiquant par partage de fichiers.
    - IHM meilleure (PC graphiques), mais fiabilité des données en très nette baisse

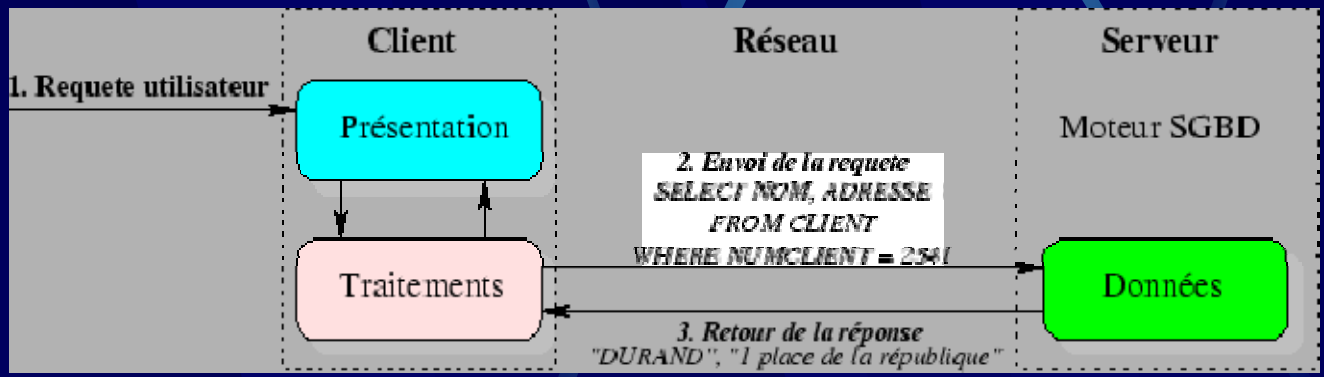
Architecture 2-tier

- Scinder les applications en 2 parties distinctes et coopérantes :
  - gestion centralisée des données (Serveur)
  - gestion locale de l'interface utilisateur (Client)



➔Ainsi est né le concept du **Client-Serveur**.

- Exemple : une application de gestion exploitant un SGBD centralisé :



- Autres exemples : Clients-Serveurs de « services classiques »
  - Mail (Thunderbird - smtp et pop3 ou imap)
  - Transfert fichiers (FileZilla - ftp), Web (FireFox - http), …

Architecture 2-tier (suite)

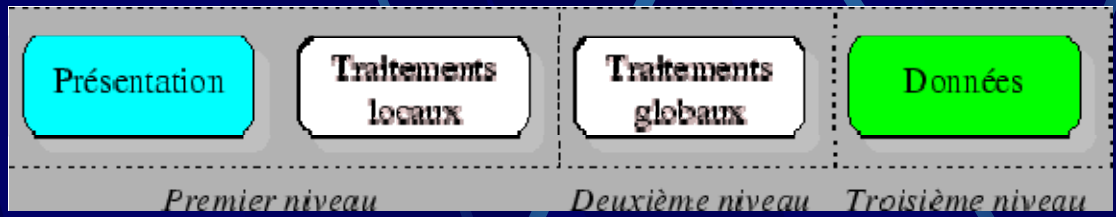
- Bilan positif :
  - IHM riche et agréable
- Bilan négatif : le client contient l'ensemble des traitements applicatifs, on dit que c'est un client « lourd », ce qui implique :
  - Problèmes de charge sur les clients
  - Problèmes de mise à jour des clients
  - Charge du réseau : beaucoup de trafic entre les clients et le serveur
  - Architecture coûteuse et complexe à maintenir

➔ Recherche d'une architecture plus évoluée, facilitant les forts déploiements à moindre coût.

➔ Réponse : les architectures distribuées (3-tier puis n-tier).

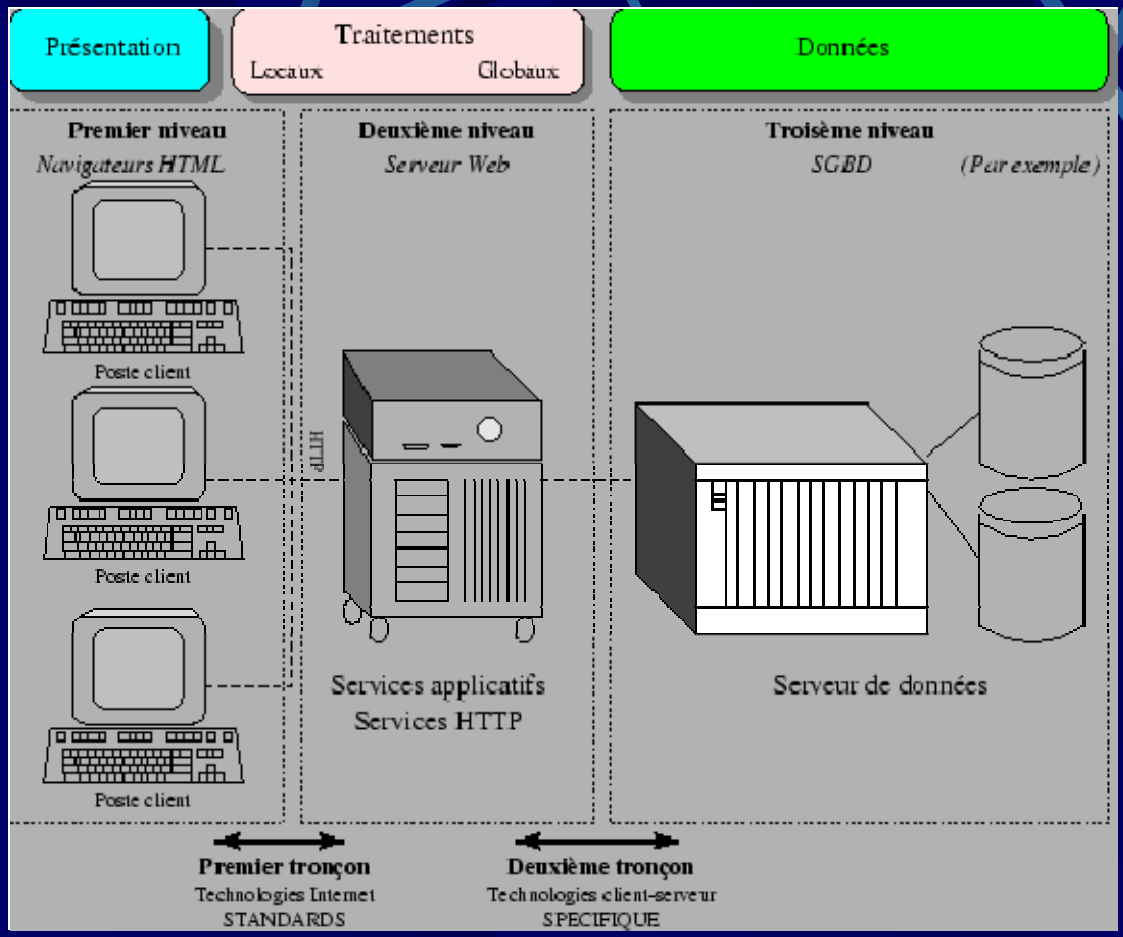
Architecture 3-tier

- Encore appelée **Client-Serveur** de 2ème génération ou Client-Serveur distribué.
- Sépare l'application en 3 niveaux de service distincts :



  - **premier niveau :** la présentation et les traitements locaux (contrôles de saisie, mise en forme de données...) sont pris en charge par le **poste client**,
  - **deuxième niveau :** les traitements applicatifs globaux sont pris en charge par un **serveur applicatif intermédiaire**,
  - **troisième niveau :** les services de base de données sont pris en charge par un **serveur de SGBD**.

Architecture 3-tier (suite) Exemple d'un intranet



- Le poste client prend la forme d'un simple navigateur Web.
- Le service applicatif est assuré par un serveur HTTP.
- La communication avec le SGBD met en oeuvre les mécanismes bien connus des applications Client-Serveur de la 1ère génération.

Architecture 3-tier (suite)

- Bilan positif :
  - Client « léger » : navigateur web interprétant du HTML ou du Java (applet)
  - Traitements applicatifs déportés sur un serveur :
    - CGI (Common Gateway Interface) : programmes écrits dans un langage quelconque (C, C++, perl, sh, …)
    - Scripts ASP ou PHP, Servlets Java
- Encore un point négatif :
  - Le serveur HTTP constitue la pierre angulaire de l'architecture : il se trouve souvent fortement sollicité.
  - On se retrouve confronté aux épineux problèmes de dimensionnement serveur et de gestion de la montée en charge.

➔ Contraintes inversées par rapport aux architectures 2-tier :
le client est soulagé, mais le serveur est fortement sollicité.

➔ Le juste équilibrage de la charge entre client et serveur semble atteint avec la génération suivante : les architectures n-tier.

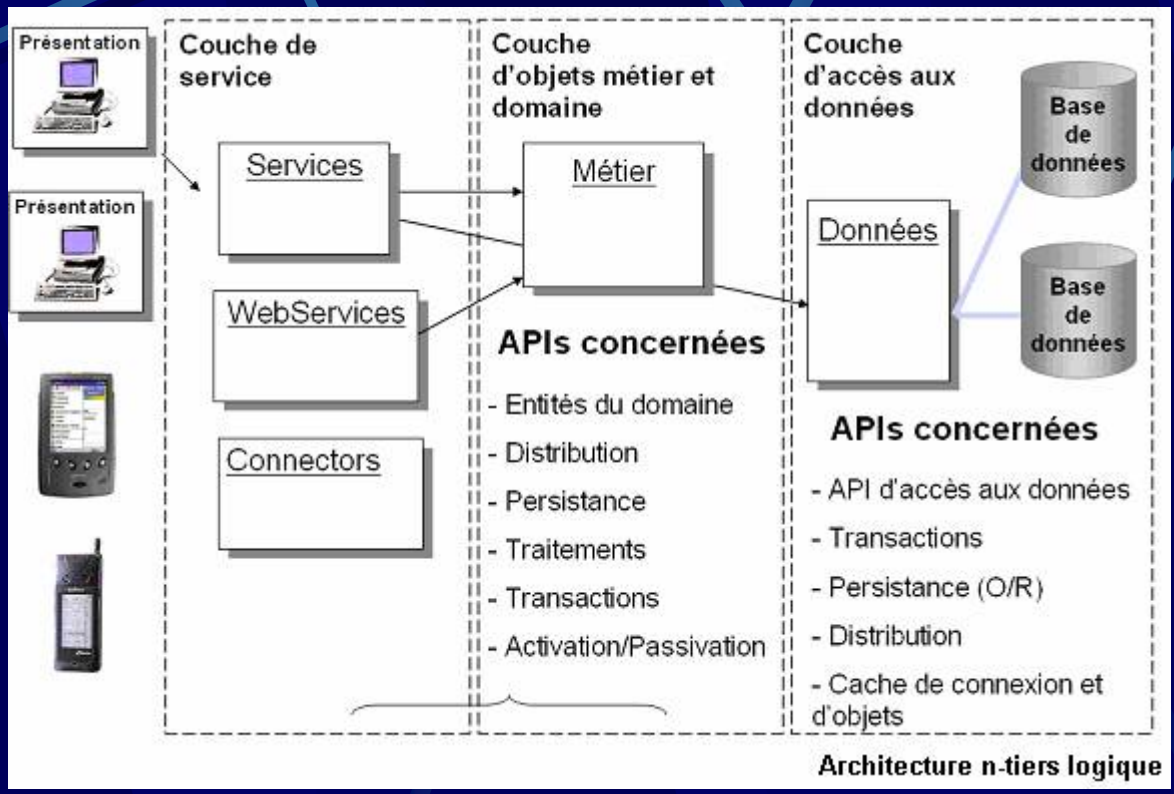
Architectures n-tier

- L'appellation « n-tier » **ne signifie pas un nombre indéterminé de niveaux** de service, ils sont toujours 3 (ou 4 selon les avis) !
  En fait, l'architecture n-tier qualifie la distribution d'application entre de multiples services et non la multiplication des niveaux de service.

- Cette distribution est facilitée par l'utilisation de **composants** ou **objets « métier »**, spécialisés, indépendants et réutilisables.
  Ils rendent un service si possible générique et clairement identifié.
  Ils sont capables de communiquer entre eux et peuvent donc coopérer en étant implantés sur des machines distinctes.

- Une architecture n-tier comprend généralement une **couche de présentation**, une **couche de services et d'objets métier** et une **couche d'accès aux données**.

Architectures n-tier (suite)

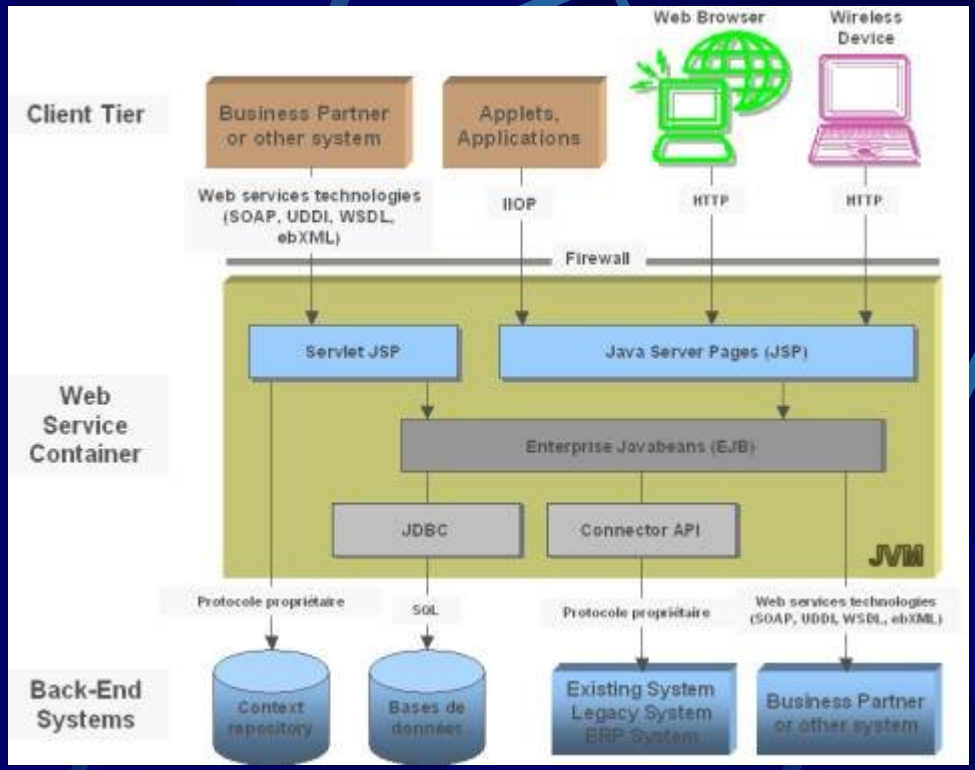


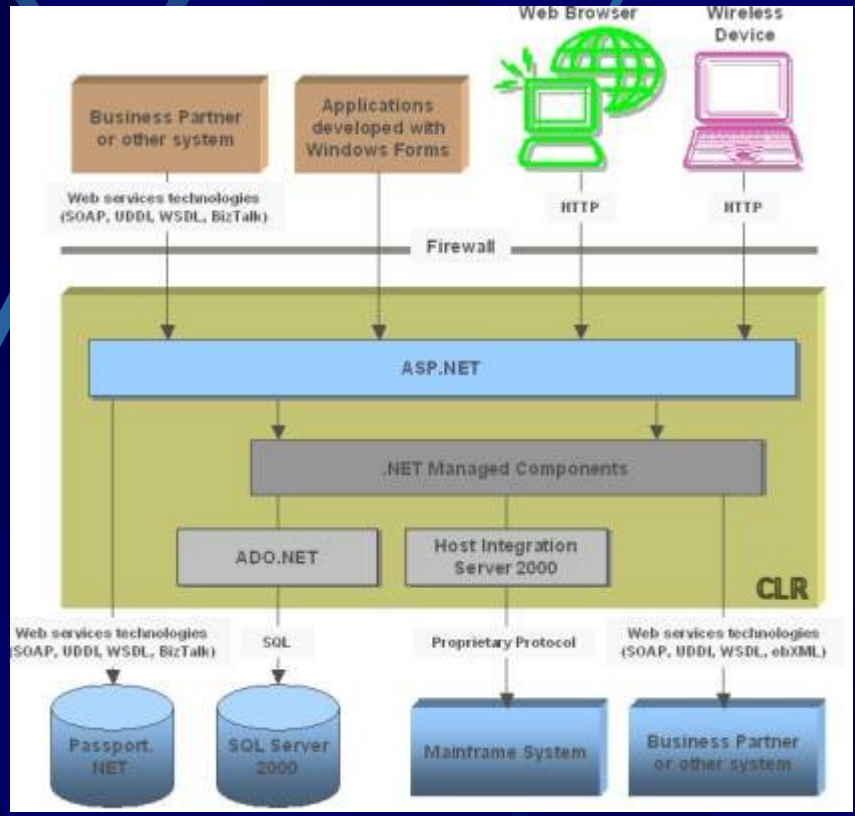
Source : http://dotnetguru.org/articles/architecturedotnet.htm

Architectures n-tier (suite)

- Couche de présentation :
  - client lourd (appli dédiée)
  - ou léger (utilisant un navigateur)
- Couche de service : traitements applicatifs
  - Besoins simples : implémentation personnelle
  - Sinon : s'appuyer sur un framework fourni
- Couche d'objets métier :
  - assure l'indépendance totale entre le client et le type de stockage utilisé (SGBDR, SGBDO, fichiers XML, ...)
  - Le client doit posséder uniquement une vue sur un objet avec l'ensemble de ces attributs (et pas un curseur pointant sur un enregistrement)
- Couche d'accès aux données ou de persistance :
  - est responsable de la création, destruction et chargement des objets métier de manière totalement transparente

J2EE et .NET





Architectures n-tier étudiées au semestre 4 au travers de deux conférences d'intervenants professionnels et des TD/TP.

Principaux objectifs du module

Principaux objectifs du module

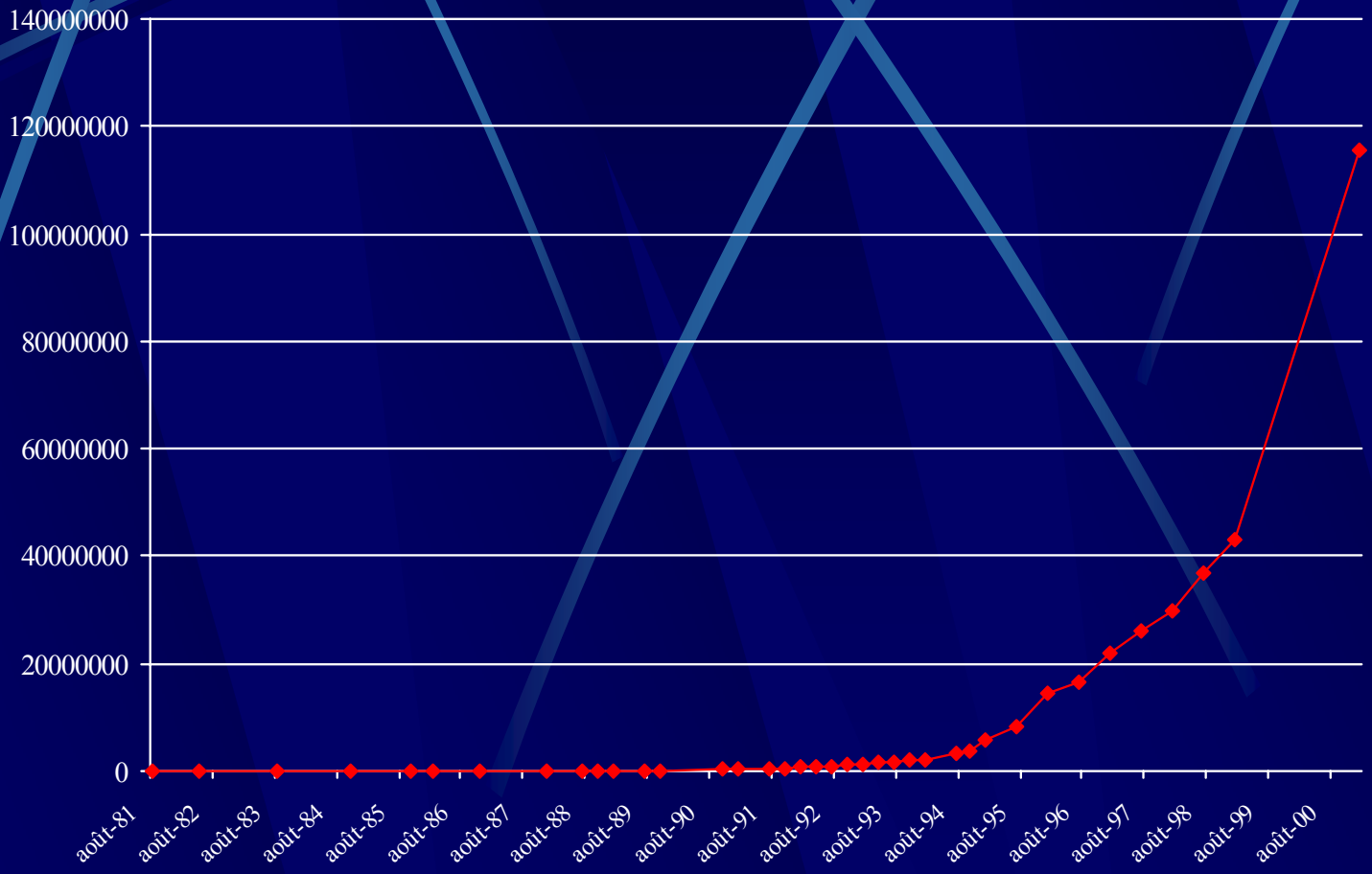
- Initiation
  - Au fonctionnement Client / Serveur du web et du protocole HTTP
  - Aux techniques de développement web « multi-niveaux » (3-tier)
- Page simple (statique) : (X)HTML, CSS
- Page dynamique :
  - Côté client : Javascript, AJAX, etc…
  - Côté serveur : PHP, ASP.NET, etc… et interaction avec un SGBD
- XML, les normes W3C, …

Architecture TCP/IP

Bref historique

- 1967 : Proposition de réseau à commutation de paquets
- 1969 : Premiers nœuds ARPANET
- 1972 : Premiers services : Telnet, FTP
- 1981 : Autres réseaux : BITNET, CSNET
- 1982 : TCP/IP choisi pour ARPANET.
  - « internet » ensemble de réseaux interconnectés, utilisant TCP/IP
  - « Internet » ensemble d'internets TCP/IP interconnectés
- 1984 : Serveurs de noms (DNS), un millier d'hôtes
- 1986 : NSFNET (backbone à 56Kbps), NNTP
  1988 : Premier virus Internet
  - Computer Emergency Response Team
- 1989 : Plus de 100,000 hôtes
- Concept web : Centre Européen Recherche Nucléaire, 1989.
- décembre 1991 : Prototype web.
- 1992 : Plus d'un million d'hôtes : Banque mondiale
- 1993 : Navigateur Mosaic.
- 1994 : Arrivée du commerce en ligne
  - Pizza-Hut, First Virtual première banque
- 1994 : Création du World Wide Web Consortium (W3C).
- 1995 : Les services en ligne proposent une connexion Internet
- Création Netscape Communication Corp. en 1995. [fin en 1998]

Une explosion
du nombre de machines sur Internet



Données et adresses

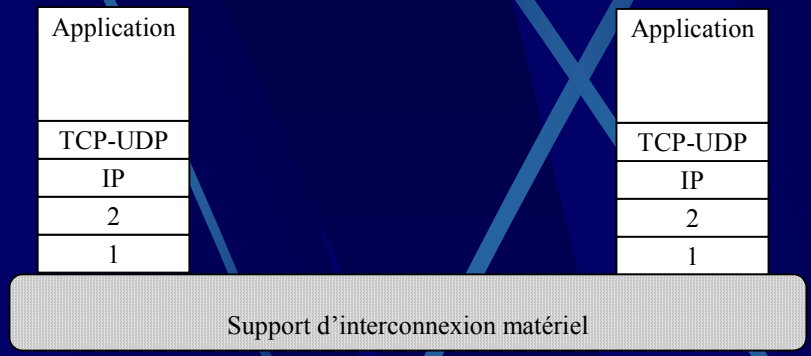
- Routage
  - Découpage des données → datagrammes.
  - Datagrammes : unités (de données du protocole IP) de taille réduite, transmises séparément sur le réseau (routés) et réorganisés à l'arrivée

- Chaque machine a une adresse Internet (IP)
  - Structure hiérarchique
  - IPV4: 4 octets (de la forme : 115.23.47.105)
    - 256^4 = 4 294 967 296 adresses disponibles (en principe)
  - IPV6: 16 octets
  - Regroupés en deux parties : (R,H)
    - R : numéro du réseau, H : numéro d'une machine dans le réseau
    - Plusieurs découpages possibles

Nom d'une machine

- (R,H) devient (nom_hôte, nom_réseau)
  - Le premier mot est le nom de la machine
  - Le nom de réseau est structuré en domaines et sous-domaines
    - e1.info.iut.u-bordeaux1.fr, ca-bdx-3-217.abo.wanadoo.fr
- Résolution de noms
  - Le réseau n'utilise que les numéros
  - Correspondance Nom → Adresse
    - Fichier hosts, Serveur de noms (Domain Name Server)

Architecture TCP/IP



Niveaux physique et liaison (couches 1 & 2) : tout est possible.

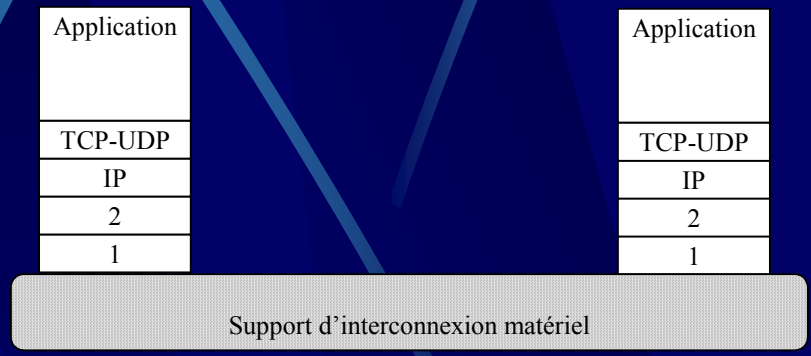
- Tous types de lien physique : coaxial, paire torsadée, modem, fibre optique, radio, …
- Tous types de couche 2 : Ethernet, FDDI, ATM, RNIS, PPP, PPTP…

Niveau réseau : IP (Internet Protocol)

- Construction des datagrammes, adressage, routage
- IP : Internet Protocol
  + d'autres (ARP, ICMP, …)

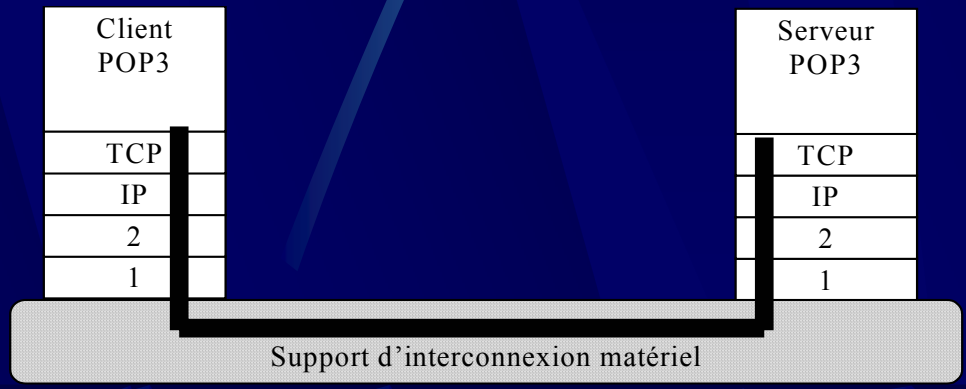
Niveau transport

- Deux protocoles
  - UDP : User Datagram Protocol
    - *Mode non connecté*, non fiable
  - TCP : Transmission Control Protocol
    - *Mode connecté*
    - Vérification de la connexion
    - Correction des erreurs
- Ports : numéros désignant des services
  - ftp 21/tcp
  - telnet 23/tcp
  - smtp 25/tcp mail
  - pop3 110/tcp post office protocol
- Le couple numéro IP + port (socket) permet d'adresser une application de façon non-ambiguë



Niveau application

- Architecture Client/Serveur
- Des services avec des protocoles précis
  - Définis par des RFC : Request For Comment (protocole, un numéro de port…)
- Les services de base
  - Connexion distante (1972 : RFC 854, port 23)
    - Un client : telnet, un serveur: telnetd (telnet daemon)
  - Transfert de fichiers (RFC 959, port 21) : File Transfer Protocol
    - Un client ftp , un serveur, ftpd
  - Courrier Electronique : SMTP (port 25), POP3 et IMAP4 (Internet Message Access Protocol),
  - **Serveurs Web : HTTP, port 80**



World Wide Web (WWW) et Protocole HTTP

Architecture World Wide Web (1)

- Un immense système d'information réparti : ensemble de documents (ou pages) web.
- Normes : consortium W3C, plus de 429 membres (nov. 2006), normes='recommandations'
- Organisation hypertexte et hypermédia (Vannevar Bush, 1945)
  - Information découpée en document (pages), reliées par des liens
  - Un clic de souris pour naviguer
- Principe de l'interaction
  - On insère une zone réactive (texte, image ou partie d'image),
  - On lui associe une action (appeler une autre page, envoyer un courrier, jouer un morceau de musique, lancer une vidéo)
  - On signale la zone (commentaire, soulignement, changement de curseur)

Architecture World Wide Web (2)

- Architecture Client/Serveur
  - Client pour naviguer sur les pages, appelé 'navigateur' (browser)
  - Serveur web pour héberger les pages

- Mécanisme de localisation des documents web (resource)
  - Types des documents très divers, essentiellement de l'hypertexte
  - Où la page se trouve-t-elle ? Comment peut-on accéder à la page ?
    ➔ URL (Universal Resource Locator)
    - Format général : `protocole://serveur:port/chemin`

    Ex : `http://dept-info.labri.fr/~felix/index.htm`
    - Types d'URL les plus courantes : http, ftp, file, mailto

    Ex : `ftp://ftp.labri.fr/pub/doc.pdf`

Protocole HTTP (1)
Protocole de transfert pour le WWW
(RFC 2616)

- Définit les messages échangés entre le client et le serveur
- Méthodes : opérations gérées par le serveur. Exemples :
  - GET : demande au serveur l'envoie d'une page (la plus utilisée)
  - PUT : permet d'écrire un ensemble de pages sur le serveur
  - POST : transmet une URL et ajoute de nouvelles données
- En-tête de message :
  - La ligne de requête peut être suivie de lignes supplémentaires : en-tête de message
  - Par exemple : L'en-tête User-Agent permet au client de fournir les caractéristiques du navigateur
  - Autres en-têtes : Accept_language (requête) pour préciser les langues naturelles que le client peut gérer, cookie (requête) pour retourner au serveur un cookie qu'il a placé au préalable, Last-Modified (réponse) pour préciser l'heure et la date auxquelles la page a été modifiée…

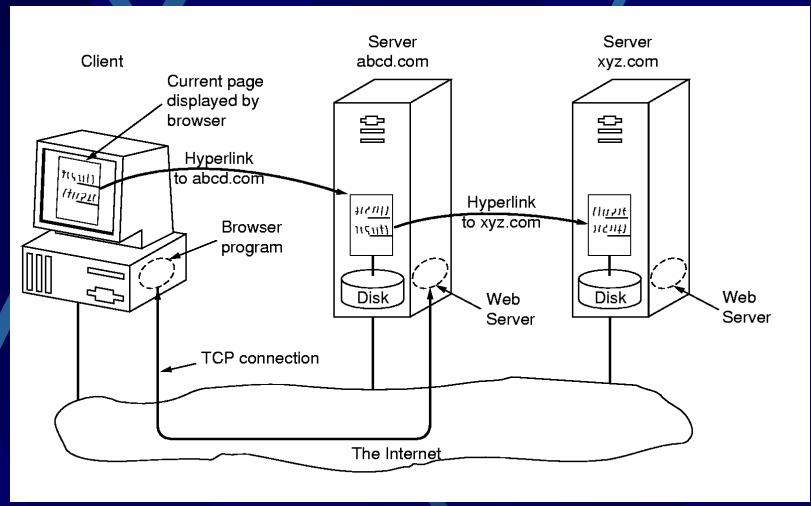
Fonctionnement côté client (1)

Principes:

1. Le navigateur détermine l'URL *http://www.labri.fr/index.html*, puis interroge le DNS pour avoir l'adresse IP *147.210.98.200.*
2. Demande connexion sur le port *80 (HTTP)*
3. Envoie une requête demandant le fichier *index.html*
4. Le serveur envoie le fichier demandé
5. La connexion TCP se termine
6. Le navigateur affiche la page (texte, image…) correspondant à *index.html*

Limitation:

1. Protocole « sans état » : pas de suivi de la connexion…



Fonctionnement côté serveur

En simplifiant :

1. Accepter une connexion TCP,
2. Obtenir le nom du fichier demandé,
3. Récupérer le fichier sur le disque et l'envoyer au client,
4. Libérer la connexion TCP

- Une requête ⇒ un accès disque (~ 5ms pour un disque SCSI)
  - 200 requêtes/seconde ⇒ faible pour un site avec beaucoup d'activités !
  - Améliorations :
    - Mémoire cache (pour conserver les fichiers les plus lus)
    - Serveur multi-thread (attention aux conflits dans les accès disque…)
    - Dupliquer serveur et disque
- Autres tâches du serveur
  - Résoudre le nom de la page web demandée (`http://www.labri.fr`)
  - Authentifier le client, puis réaliser un contrôle d'accès sur le client et sur la page (fichier `htaccess`)
  - Vérifier le cache avant de rechercher la page sur le disque
  - Déterminer le type MIME pour l'inclure dans la réponse au client
  - Enregistrer l'activité réalisée dans le journal

Amélioration des performances

- WWW est le service le plus populaire : surcharge des serveurs, routeurs, liaisons…
- Techniques pour améliorer les performances :
  - Mise en cache
    - Mémorisations des pages en vue d'un emploi ultérieur
    - Un processus 'proxy' gère la mise en cache
    - Navigateur configuré pour adresser les requêtes au proxy (et non au serveur hébergeant la page)
      - Les pc, les LAN (d'entreprise) et les FAI disposent d'un 'proxy'
      - Mise en cache hiérarchique
      - Pb : combien de temps les pages doivent-elles rester en cache ?
        - Utilisation de l'en-tête '*Last-Modified* ' : stabilité d'une page…
        - Envoi par le proxy de l'en-tête '*if-modified-since*' : réponse courte du serveur si la page n'est pas périmée.
        - Pages dynamique (serveur) ne doivent pas être mises en cache.

Amélioration des performances

- Techniques pour améliorer les performances (suite)
  - Réplication de serveur (ou mise en miroir)
    - Contrairement à la mise en cache, ce procédé est une technique mise en œuvre côté serveur
    - Difficulté d'anticiper sur le succès d'un site, parfois le succès n'est que temporaire (flash crowd : afflux soudain de visiteurs)
    - Réplication automatique en fonction du trafic
    - Service proposé par des sociétés qui disposent de plusieurs sites d'hébergement
  - Réseaux de remise de contenu
    - Sociétés (CDN : Content Delivery Network) proposant de distribuer des contenus rapidement et facilement.
    - CDN : Interface entre fournisseurs de contenu (sites de musique, vidéo…) et FAI

Mise en oeuvre

- Serveurs
  - Produits disponibles :
    - Apache
    - Lotus Domino Web Server
    - Microsoft I.I.S.
    - Netscape Enterprise Server
    - Oracle Web server
  - Caractéristiques :
    - Robustesse, fiable, sécurité…
    - Communiquer avec des SGBD
  - Critères de choix :
    - Système d'exploitation
    - SGBD associé
- Navigateurs
  - Produits disponibles :
    - Mozilla FireFox
    - Internet Explorer (MS)
    - Opera
    - Safari
  - Récents et efficaces
    - Pour être capable d'exploiter toutes les évolutions

Documents web statiques

HTML

- HTML (Hyper Text Markup Language) :
  - Un langage pour décrire le texte de la page (contenu/données) et les informations de formatage (présentation/mise en forme)
  - Formatage réalisé à l'aide de balises (tags) indiquant au navigateur comment il doit afficher la page
    - Langage de balisage (comme TeX, troff…)
    - Une balise est entourée de : < et >, </ et >, < et />
    - Exemple : niveaux de titre, centrage, tableau, listes de données

      `<html> </html>, <h2> Titre de niveau 2 </h2>, <br/>`
- Normes : consortium W3C, plus de 396 membres (nov. 05), normes='recommandations'
- Mécanisme de localisation des documents web (resource)
  - Types des documents très divers, essentiellement de l'hypertexte
  - Quel est le nom de la page ? Où la page se trouve-t-elle ? Comment peut-on accéder à la page ?
    - URL (Universal Resource Locator) :
      - Format général : `protocole://serveur:port/chemin`

        `http://www.labri.fr/~felix/enseignement/index.html`
      - Types d'URL les plus courantes : http, ftp, file, mailto

        `ftp://ftp.labri.fr/pub/doc.pdf`

De HTML à XHTML 2.0

HTML (89/90) : les débuts

HTML + (93) : liens, images, listes (unidirectionnel)

HTML 2.0 (nov. 95) : images cliquables, formulaires

- Balise `<input>`
  - Champs de texte, case à cocher, image réactive, bouton de soumission…
- Exemple de formulaire avec méthode `post` : code source, page formatée et commentaire

HTML 3.2 (janv. 97) : tableaux

- Balise `<table>`

HTML 4.01 (déc. 99) : scripts

XHTML 1.0 : « version XML » de HTML 4.01

XHTML 1.1 : Modularisation de XHTML

XHTML 2.0 : en cours de développement… Malheureusement non compatible avec les versions précédentes !

XHTML 1.0 (eXtended HTML)

- Un langage avec une syntaxe stricte
  - Analyseur plus simple (donc adapté à des équipements légers)
  - Reformulation de HTML permettant la structuration du contenu
  - Permet la séparation contenu / présentation
    - La balise `<h1>` n'a plus de signification intrinsèque : le formatage est à définir dans un fichier XSL.

- Différences HTML 4.01 et XHTML 1.0
  1. Balises et attributs en minuscule
  2. Balises de fermeture nécessaires
     (`<p> </p>`, `<img src="Image1.jpg"/>`, `<br />`)
  3. Valeurs des attributs entre guillemets
     (`<img src="i.jpg" height="500"/>`)
  4. Les balises doivent s'imbriquer correctement

Quelques balises (X)HTML

`<html>...</ html>`	Début et fin de fichier Html
`<head>...</head>`	Zone d'en-tête d'un fichier Html
`<title>...</title>`	Titre affiché par le browser (dans head)
`<body>...</body>`	Début et fin du corps du fichier Html
`<p>...</p>`	Nouveau paragraphe
`<img src="xyz.gif" />`	Insertion d'une image
`<a href="http://...">...</a>`	Lien vers une page Web
`<a href="mailto:...">...</a>`	Lien vers une adresse eMail
`<a href="fichier.htm">...</a>`	Lien vers la page locale fichier
` `	A la ligne
…	
`<!--...-->`	Commentaire ignoré par le navigateur

Deux groupes de balises

- Balises de **type bloc** (Ex : div, p, h1...h6, ul, ol, li, table, pre, …)
- Balises de **type en-ligne** (Ex : span, a, em, strong, img, br, input, …)

- Positionnement :
  - les "blocs" se placent toujours l'un en dessous de l'autre par défaut.
  - les "en-ligne" se placent toujours l'un à côté de l'autre afin de rester dans le texte.

- Imbrications et dimensions :
  - Une "bloc" peut contenir une (ou plusieurs) "bloc" et/ou "en-ligne", sauf exceptions, et avoir une dimension (largeur, hauteur définies).
  - Une "en-ligne " ne peut contenir QUE une (ou plusieurs autres) "en-ligne".

Structure d'un document XHTML: 3 parties

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"  lang="fr">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Texte du titre de la page</title>
</head>

<body>
Contenu …
</body>

</html>
```

DOCTYPE : 3 variantes

"Strict", "Transitional", et "Frameset"

Pour HTML 4.01 :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
       "http://www.w3.org/TR/html4/strict.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "http://www.w3.org/TR/html4/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
       "http://www.w3.org/TR/html4/frameset.dtd">
```

Pour XHTML 1.0 :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Pour XHTML 1.1 : une seule variante !

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Exemple de fichier XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"  lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ma première page web</title>
</head>
<body>
<h1>Ceci est un titre</h1>
<p>Voici un paragraphe pas trop long</p>
<h1>Encore un titre</h1>
<p>… et un autre paragraphe</p>
</body>
</html>
```

Feuilles de style en cascade CSS (Cascading Style Sheet)

- Permettent une stricte séparation du contenu XHTML et des informations de mise en page
- But : même apparence pour toutes les pages d'un même site
  - Eviter les styles « en dur » (<b>, <i>,…) ou en attribut de balise
  - Utiliser des styles logiques définis dans la feuille de style

- Principe :
  - Associer des paramètres de mise en page à certaines balises
  - Ces paramètres se propagent en cascade d'un élément à ses fils
- Syntaxe :

```
sélecteur {règle 1; règle 2; ... }
```

Exemple :

```
h1 {color: red; font-style: italic; border-width: 1;
   border: solid; text-align: center;}
```

Insérer un style : 3 possibilités

- Description directement dans le document (en-tête) :

```
<head>
 <style type="text/css">
   h1 {border-width: 1; border: solid; text-align: center}
 </style>
</head>
<body>
 <h1> Cet h1 est affecté par notre style</h1>
</body>
```

- Description dans un fichier externe et référence avec une balise `link`

```
<link rel="stylesheet" type="text/css" ref="../Style.css">
```

- Style défini « inline »

```
<h1 style="color=blue"> Un élément h1 en bleu </h1>
```

Exemple de fichier XHTML + CSS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"  lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ma première page web</title>
<style type="text/css">
    h1 {color: red; font-style: italic; border-width: 1; border: solid; text-align: center;}
</style>
</head>
<body>
<h1>Ceci est un titre</h1>
<p>Voici un paragraphe pas trop long</p>
<h1>Encore un titre</h1>
<p>… et un autre paragraphe</p>
</body>
</html>
```

Sélecteurs

- Le sélecteur *class* : un nom précédé d'un point

```
CSS   : .rouge { color: red; }
XHTML : <p class="rouge">ça s'affiche en rouge</p>
```

  - ➔ Cette mise en forme peut être répétée autant de fois qu'on veut.

- Le sélecteur *id* : un nom précédé d'un #

```
CSS   : #contenu { margin-left:110px; }
XHTML : <div id="contenu">blabla</div>
```

  - ➔ presque la même fonction que *class*, à la différence importante qu'on ne peut l'utiliser **qu'une seule fois dans la page,**
  - ➔ plutôt utilisé à la mise en page qu'à la mise en forme de caractères.

Visualiser un exemple complet

Compléments

➔ Consulter l'aide mémoire CSS
aide-memoire_css.pdf

Puissance des CSS

- Le site de Zen Garden : http://www.csszengarden.com
- Le site d'Alsacréations : http://www.alsacreations.com
  - Les gabarits de mise en page CSS
    http://css.alsacreations.com/Modeles-de-mise-en-page-en-CSS
  - La galerie de menus CSS
    http://css.alsacreations.com/Galeries-de-menus-en-CSS

Développement web

De la page au site web

- A la main
  - En saisissant le texte et les diverses balises
  - Long et fastidieux … mais on maîtrise le contenu exact !
    - Editeurs de texte : Emacs, NotePad++, …
- A l'aide d'outils adaptés
  - Suites bureautiques
    - MS Office par exemple (Word, mais aussi Excel, Access ou Powerpoint)
  - Outils spécialisés
    - pour créer une page ou développer un site complet

      Microsoft Expression Web, Macromedia Dreamweaver, Adobe GoLive, Nvu (logiciel libre), Microsoft Visual Studio, ...
  - Système de gestion de contenu « CMS »

Gestion d'un site

- Développement des pages sur un serveur local
  - Serveur de fichier sauvegardé
  - Serveur Web de test
- Publication des pages
  - Le serveur Web est à l'extérieur du firewall
  - Les pages sont publiées par FTP, WebDav, ...
- Les pages sont testées
  - Localement
  - En situation réelle
  - En utilisant plusieurs types de navigateurs

A garder à l'esprit…

- On ne connaît pas la nature du client
  - Système : PC sous Windows, MacIntosh, station Unix
  - Navigateur et sa version : FireFox, Explorer, ...
- Tous doivent être à égalité
  - La page doit être lisible par tous
  - Les programmes exécutables par tous
  - Les données transférées le moins volumineuses possible
- Une validation W3C est la bienvenue !!
  - XHTML : http://validator.w3.org
  - CSS : http://jigsaw.w3.org/css-validator