Systèmes d'Information Distribués

Université Lyon 1
UFR informatique

L. Médini, mars 2008

Plan des cours

- Langages de description de l'information
  - Langages descriptifs XML et XHTML
  - Langages de transformation
  - Programmation avec XML
- Protocole HTTP et programmation côté serveur
- Architectures réparties (CORBA, RMI)
- Objets distribués (Javabeans, EJB)
- Web services (SOA, WSDL, SOAP, UDDI)
- Projet

Introduction

- De quoi on va parler dans ce cours ?
  - De « Nouvelles Technologies de l'Information et de la Communication »
    - Langages dédiés au web (XHTML, XML...)
    - Programmation web
- Points de vue abordés
  - Structuration des données
  - Échange et partage des données
- Ce qu'on ne fera pas
  - Applications web (« services Web »)
  - Aspects sémantiques (« Web sémantique »)

Introduction

- Définitions
  - L'information en tant que signal circulant entre un émetteur et un récepteur lors d'un processus de communication
    - Origine : théorie de l'information (Shannon 1948), traitement du signal (Fourier ~1800)
    - On s'intéresse uniquement à l'aspect physique du signal transmis
    - Émetteur et récepteur peuvent être humains ou logiciels.

Introduction

- Définitions
  - Donnée : valeur associée à un type de données
    - Décrite par des caractéristiques de forme
    - Indépendante de son interprétation : donnée « brute »
  - Élément d'information : ensemble de données faisant « sens »
    - On parle aussi de « grains d'information »
  - Document : regroupement cohérent d'éléments d'information
    - Autour d'une thématique commune
    - Selon une structure donnée

Introduction

- Définitions
  - Information structurée : bases de données
    - Éléments d'information stockés séparément
    - Accès par requêtes
    - Traitements facilités
  - Information non structurée : corpus documentaires
    - Éléments d'information stockés sous forme de textes
    - Accès par recherche d'information
    - Traitements complexes
  - Information semi-structurée : langages à balises
    - Éléments d'information stockés dans des documents
    - Permet les deux types d'accès et de traitements

Introduction

- Définitions
  - Balise : signe marquant une position particulière
    - Signe = élément d'information
    - Marquer = permettre la distinction
      - Ex : fusée de détresse, signal radio, « tag » HTML
    - Position = par rapport à l'espace considéré
      - Ex : espace 3D (avion), 2D (bateau), 1D (document)
    - Particulière = en général, la position d'un élément de l'espace qu'on cherche à repérer
      - Ex : aéroport, bateau, élément d'information

⇒ Type d'information facilement repérable permettant d'identifier d'autres éléments informationnels (i.e. « méta-information »)

Introduction

- Définitions
  - Balisage documentaire : utilisation de balises
    - Pour marquer des points précis d'un document
    - Pour marquer des zones (segments) de document
      - Balisage de début et de fin de zone
    - Pour structurer le document
  - ⇒ Utilisation de plusieurs types de balises
    - ⇒ En fonction du type d'élément à marquer
    - ⇒ En fonction du type de marquage (point, début, fin)
  - Langage à balises : langage de description de documents utilisant ces techniques de balisage
  - Élément : une balise et son contenu

Introduction

- Quelques langages à balises
  - Langages de *balisage procédural* : permettent de décrire la mise en forme (formatage) d'un document
    - Ex : PS, RTF, TeX, HTML
  - Langages de *balisage descriptif* : se contentent de décrire les données, sans but de traitement
    - Ex : RDF, OWL, MathML
  - *Méta-langages* de balisage : permettent de définir des langages de balisage
    - Ex : SGML, XML

Introduction

- Le langage Postscript

```
%!PS-Adobe-3.0
%%Title: Microsoft Word - Document1
%%Creator: PSCRIPT.DRV Version 4.0
%%CreationDate: 03/02/02 10:47:00
...
%%EndComments
%%BeginProlog
%%BeginProcSet: Pscript_Res_Emul 1.0 0
/defineresource where{pop}{userdict begin/defineresource{userdict/Resources 2
...
}ifelse}bind readonly def end}ifelse
%%EndProcSet
```

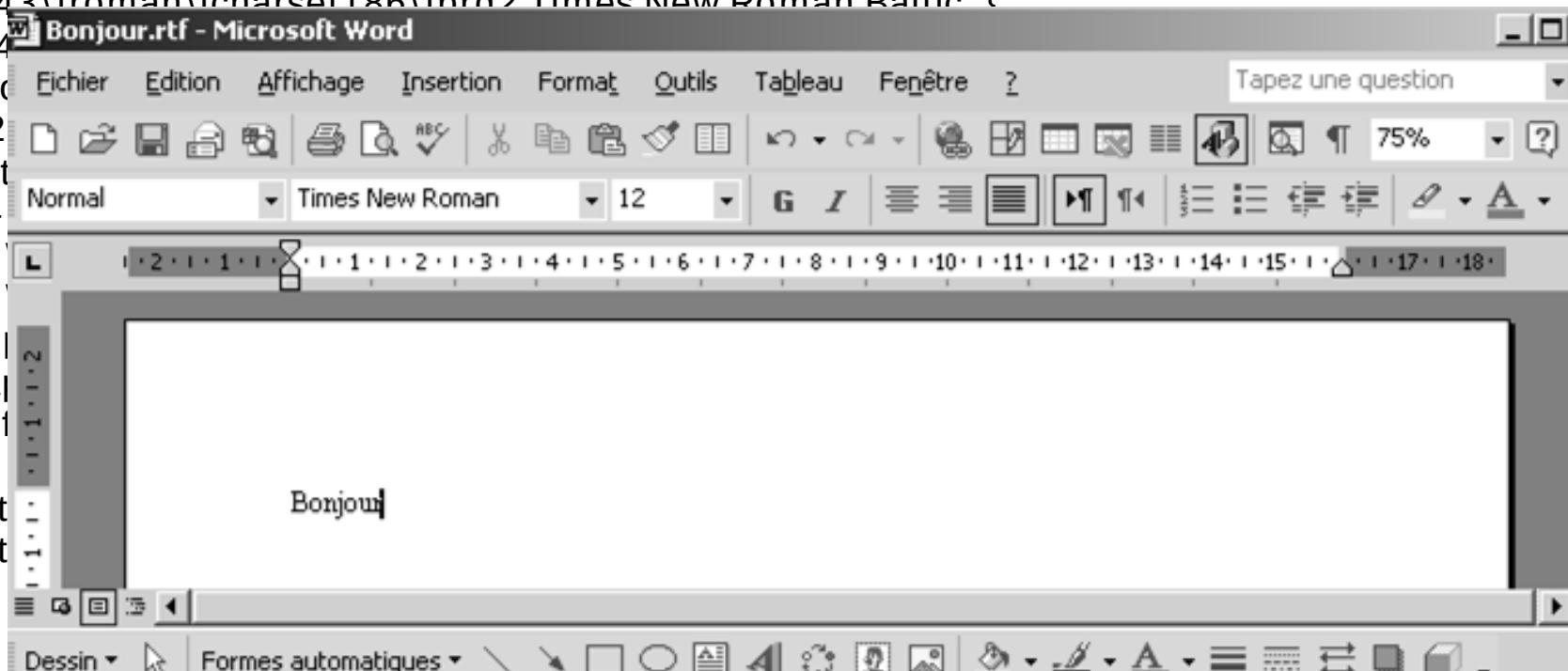
```
%%BeginResource: file
...
%%EndResource
...
%%EndProlog
%%BeginSetup
...
%%BeginFeature: *PageSize A4
...
%%EndFeature
...
%%EndSetup
%%Page: 1 1
%%BeginPageSetup
...
%%EndPageSetup
...
```

```
%%IncludeFont: Courier
...
 (Courier) cvn /Type1
...
(Essai impression)S
...
(%%[ Page: 1 ]%%) =
%%PageTrailer
%%Trailer
%%DocumentNeededFonts:
%%DocumentSuppliedFonts:
/Pscript_Win_Driver /ProcSet findresource dup /terminate g
exec
Pscript_Win_Compat dup /terminate get exec
%%Pages: 1
(%%[ LastPage ]%%) =
%%EOF
```

Introduction

- Le langage RTF

{\rtf1\adeflang1025\ansi\ansicpg1252\uc1\adeff0\deff0\stshfdbch0\stshfloch0\stshfhich0
\stshfbi0\deflang1036\deflangfe1036{\fonttbl{\f0\froman\fcharset0\fprq2{*\panose
020206030504050203040}Times New Roman;}{\f36\froman\fcharset238\fprq2 Times
New Roman CE;}{\f37\froman\fcharset204\fprq2 Times New Roman Cyr;}...
{\f43\froman\fcharset186\fprq2 Times New Roman Baltic;}
{\f4
{\cc
en2
{\st
...;}
{*
{*
{\op
\pa
\de
...
{\rt
{\rt



Introduction

- Les langages dédiés au Web
  - Historique 1/3
    - 1960-1986 : SGML (norme ISO)
    - 1989 : ODA (norme ISO, concurrent de SGML)
    - Fin des années 80 : apparition/essor du web
    - 1992-1997 : HTML (versions 1.0 -> 4.01)
    - Octobre 1994 : création du World Wide Web Consortium (W3C) : http://www.w3.org
    - 1996-1999 : CSS Level 1 (fonctionnalités de base)
    - Février 1998 : XML version 1.0
    - 1998 : CSS Level 2 (fonctionnalités supplémentaires)
    - Octobre 1998 DOM Level 1 (supporte XML et HTML)

Introduction

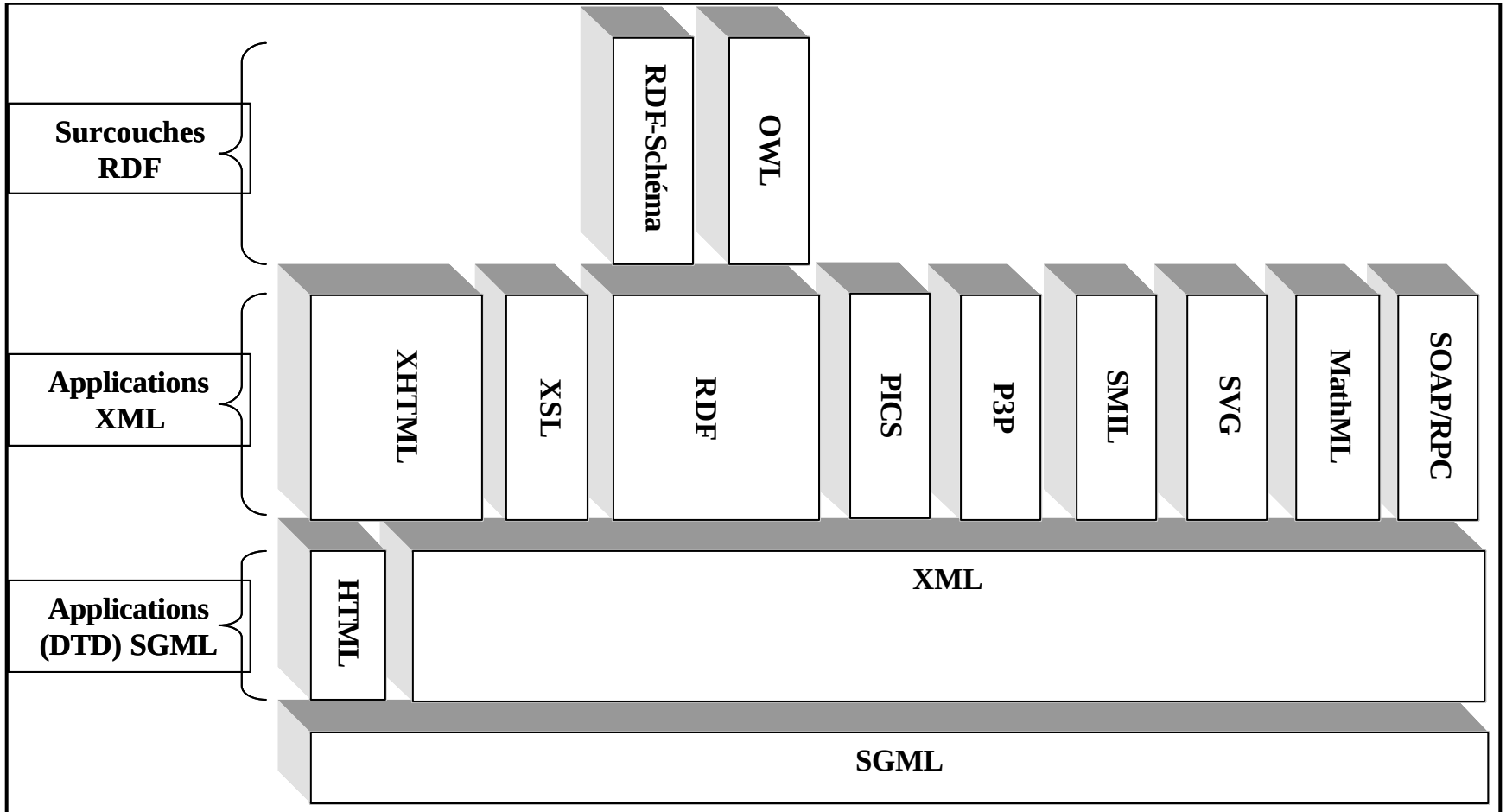
- Les langages dédiés au Web
  - Historique 2/3
    - Décembre 1999 : XHTML 1.0
    - 1999-2004 : RDF et RDF-Schema 1.0
    - Novembre 2000 : DOM Level 2 (supporte CSS et espace de noms XML)
    - Mai 2001 : schémas XML 1.0
    - Juin 2001 : XLink 1.0
    - Juillet 2001 : SVG 1.0
    - Novembre 2001 : XSL 1.0
    - Janvier 2003 : SVG 1.1

Introduction

- Les langages dédiés au Web
  - Historique 3/3
    - Février 2004 et août 2006 : XML 1.1
    - Février 2004 : OWL 1.0
    - Avril 2004 : DOM Level 3 Core
    - Octobre 2004 : XML Schema (2e edition)
    - Octobre 2004 : XQuery 1.0
    - Octobre 2004 : XPath 2.0 (Working Draft)
    - Novembre 2004 : XSLT 2.0 (Working Draft)
    - Décembre 2004 : WSDL 2.0
    - Restent en développement : CSS L3, DOM L3...

Introduction

- Quelques langages dédiés au Web



Partie 1

- Les langages de structuration de l'information dédiés au web
  - XML
  - XHTML

HTML, XML et XHTML

- HTML
  - DTD SGML : ensemble (dé)fini d'éléments SGML
  - Destiné à visualiser des hyperdocuments sur écran
  - Interprété côté client par les navigateurs
  - V. 1.0 : 1992 -> V. 4.01 (dernière) : oct. 1997
  - Souvent associé à d'autres langages
    - CSS : langage de feuilles de style
    - Langages de scripts : JavaScript, JScript, VBscript…
    - Inclusion d'objets définis dans d'autres langages : applets Java, contrôles ActiveX…
  - Remarque : le langage D(H)TML n'existe pas

HTML, XML et XHTML

- XML : principes de base
  - DTD de SGML (beaucoup plus concis)
  - Restrictions de syntaxe et non de contenu
  - Au même titre que SGML, c'est un méta-langage de description des données
    - ⇒ Ça ne sert à rien
    - ⇒ Ça permet de définir des « applications » pour faire ce qu'on veut avec
      - Visualiser des pages web
      - Décrire des images
      - Échanger des données techniques…
  - V 1.0 : fév. 1998 -> 04 fév. 2004 (3ème édition)
  - V 1.1 : 04 fév. 2004 ; 16 août 2006 (2ème éd.)

HTML, XML et XHTML

- XML : composants
  - XML (syntaxe) : documents « bien formés »
  - DTD/schémas XML : documents « valides »
  - Processeur (parser) XML : analyse et traitement
  - DOM : modèle arborescent des données d'un document pour un langage de programmation
  - SAX : accès « simple » aux données (programmation événementielle)
  - Espaces de noms (« namespaces ») : interopérabilité des applications
  - XBase, XPointer, XLink : mécanismes de liens
  - XSL : mécanismes de transformation

HTML, XML et XHTML

◘ Validation de documents XML : DTD *vs* Schémas

Caractéristique	DTD	Schémas
Syntaxe	Notation EBNF + pseudo-XML	XML 1.0
Outils	Outils SGML existants (chers et complexes)	Tous les outils XML existants et à venir
Supports DOM/SAX	Non	Oui (comme pour les fichiers XML).
Modèles de contenu	- Listes : ordonnées ou de choix - Cardinalité : 0, 1 ou plusieurs occurrences - Pas d'éléments nommés ou de groupes d'attributs.	- Listes : ordonnées et de choix (détails de contenus mixtes) - cardinalité : spécification d'un nombre exact d'occurrences possible - groupes de modèles nommés
Typage des données	Faible (chaînes, jetons nominaux, ID…)	Fort (nombres, chaînes, date/heure, booléen, structures…)
Héritage	Non	Oui
Extensibilité	Non (pas sans modification de la recommandation XML)	Oui (puisque fondés sur l'extensibilité de XML)
Contraintes légales	Compatibilité avec SGML	Aucune (simplement des « emprunts » aux DTD, comme pour les types de données)
Nombre de vocabulaires supportés	Une seule DTD par document	Autant que nécessaire (grâce aux espaces de noms)
Dynamicité	Aucune : les DTD sont en lecture seule	Peuvent être modifiés dynamiquement

HTML, XML et XHTML

- Syntaxe XML (voir poly p. 50) : un document XML bien formé est composé
  - D'un prologue, contenant :
    - Éventuellement une déclaration XML

```
<?xml version="1.0" encoding="ISO-8859-1"
  standalone="yes"?>
```

    - Éventuellement une déclaration de type de document

```
<!DOCTYPE Nom_de_l_élément_racine Type_de_source
  emplacement1 emplacement2 [sous-ensemble interne
  de DTD]>
```

  - D'un élément « racine », composé :
    - Soit d'une balise ouvrante, éventuellement d'un contenu et d'une balise fermante

```
<NomElement attribut1="valeur1" attribut2="valeur2"...>
  contenu
</NomElement>
```

HTML, XML et XHTML

- Syntaxe XML (voir poly p. 50) : un document XML bien formé est composé
  - D'un prologue, contenant :
    - Éventuellement une déclaration XML

```
<?xml version="1.0" encoding="ISO-8859-1"
  standalone="yes"?>
```

    - Éventuellement une déclaration de type de document

```
<!DOCTYPE Nom_de_l_élément_racine Type_de_source
  emplacement1 emplacement2 [sous-ensemble interne
  de DTD]>
```

  - D'un élément « racine », composé :
    - Soit d'une balise d'élément vide

```
<Nom_element_vide att1="val1" att2="val2"... />
```

HTML, XML et XHTML

- Syntaxe XML (voir poly p. 50) : remarques
  - Structure
    - Un contenu est composé de texte et / ou d'autres éléments appelés fils de l'élément courant
    - Le texte est appelé « données caractères analysables » ou PCDATA (pour Parsed Character DATA).
    - Le contenu autorisé pour le PCDATA dépend du type d'encodage choisi
    - Le fait d'avoir un unique élément racine, des éléments fils, eux-mêmes décomposables, etc. définit une **structure arborescente**
    - Pas de chevauchement de balises entre un élément père et un élément fils

HTML, XML et XHTML

- Syntaxe XML (voir poly p. 50) : remarques
  - Caractères spéciaux
    - Les caractères "<" (inférieur) et "&" (esperluète) sont interdits dans les contenus. On aura recours aux entités "<" et "&".
    - L'usage de ">" (supérieur) ou des guillemets simples ou doubles peut également être perturbant. Dans ce cas, on a recours à ">", "'" et """.
    - Si l'on veut vraiment utiliser les caractères "<" ou "&", il est possible de définir une balise sous forme de zone de caractères non analysés (CDATA), sous la forme :

```
<![CDATA [ texte comprenant des caractères
  interdits ]]>
```

HTML, XML et XHTML

- XHTML
  - DTD XML : ensemble (dé)fini de balises XML
  - But : réécrire HTML, mais en plus propre
  - V. 1.0 : janv. 2000, révisée en 2002
  - V. 1.1 : mai 2001 (pas de rapport avec XML 1.1)
  - Trois recommandations
    - XHTML frameset
    - XHTML transitionnel
    - XHTML strict

  ⇒ Séparation du fond et de la forme

  ⇒ Pris en charge par tous les navigateurs récents

  ⇒ Pris en charge par les outils de traitement XML

HTML, XML et XHTML

- Syntaxe (X)HTML :
  - Polycopié : page 21
  - Tutoriel web : http://pci.univ-lyon1.fr/TP/Tutoriel-TP11/

Partie 2

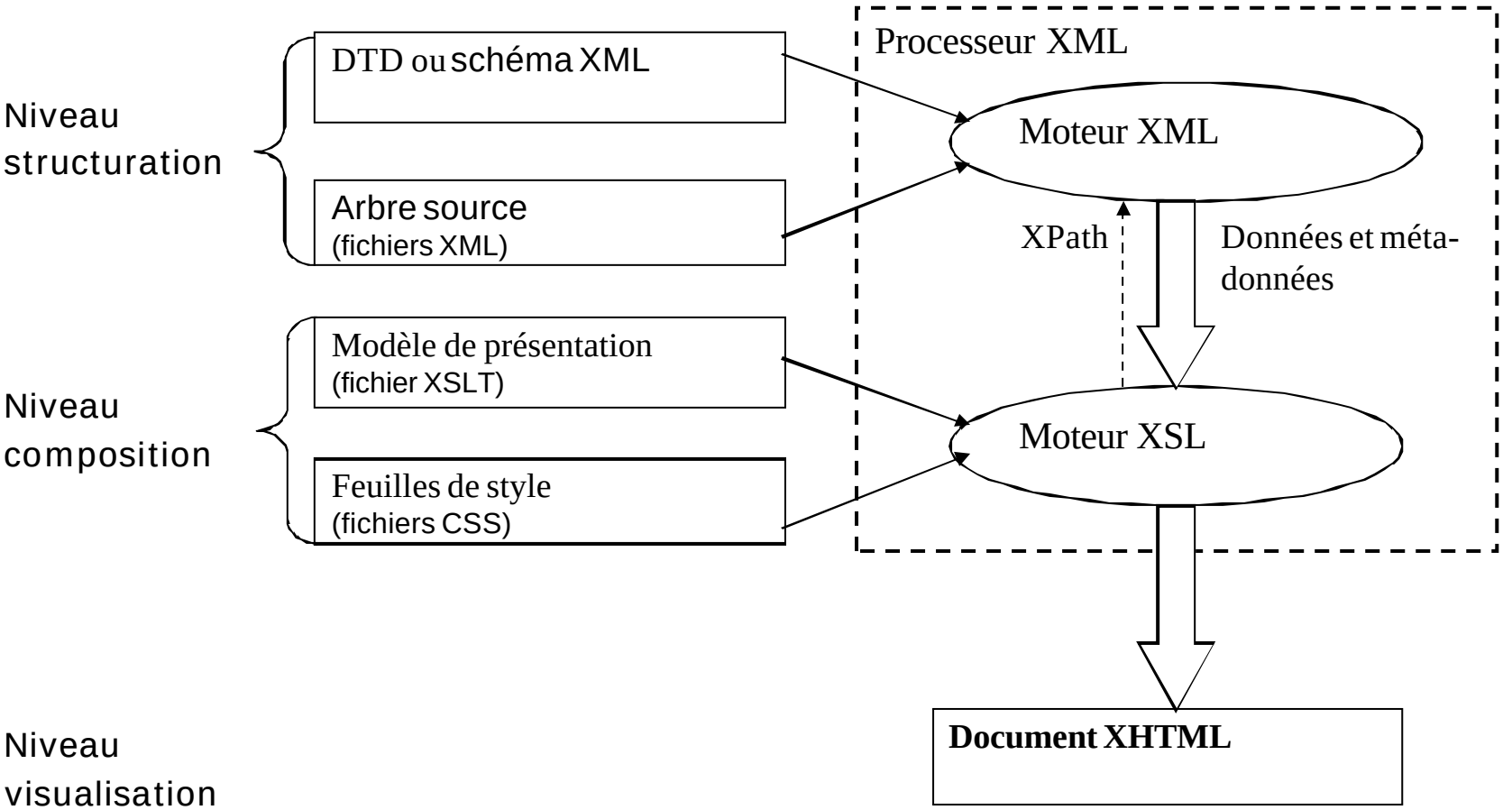
- Les langages de transformation et de mise en forme
  - XSL
  - CSS

Transformation d'arbres XML : XSL

- Caractéristiques de XSL
  - Officiellement : XML Stylesheet Language
  - En pratique, ça ne sert à rien d'appliquer des éléments de style à un document XML
  - Mais XSL fournit un mécanisme très puissant pour transformer un arbre XML
    - En un autre arbre XML (échange de données)
    - En un arbre XHTML (visualisation des données XML)
    - En un texte simple (fichier non destiné à une application utilisant un parser XML)
    - En un document papier formatté (XSL-FO)

Transformation d'arbres XML : XSL

- Utilisation la plus courante de XSL



Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Permet de pointer vers les données de l'arbre XML
      - pour le parcours de documents XML
      - pour le test de valeurs associées aux contenus ou aux attributs d'éléments
    - Ne respecte pas la syntaxe XML
      - pour ne pas « perturber » l'analyse des feuilles de style XSLT par le parser XML

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Nœud
      - Tout type de données (élément, attribut, PI)
      - Racine du document : '/'
      - Les éléments sont identifiés par leurs noms
      - Les attributs sont identifiés par '@' suivi du nom de l'attribut
    - Chemin de localisation
      - Absolu : à partir de la racine de l'arbre XPath
      - Relatif : à partir du nœud contextuel
      - Récursif : à partir du nœud contextuel, mais seulement « vers le bas »

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Axes de navigation
      - Déplacements complexes dans l'arbre XPath
      - Voir liste poly p. 75
    - Filtrage des nœuds
      - Permet de sélectionner des nœuds par leurs contenus
      - Définition des caractéristiques recherchées entre crochets
      - Opérateurs de comparaison simples
      - Fonctions XPath
        - Expression de caractéristiques de sélection complexes
        - Voir liste poly p. 74

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : principes de base
    - Description de l'arbre résultant (programmation déclarative)
    - Application XML définissant des « éléments de transformation »

      ⇒Référence à un espace de noms spécifique « `xsl:` »
    - Balises spécifiques interprétées par un processeur XSLT
    - Structuration par modèles (« templates ») de contenus
      - Définissant le traitement à appliquer à un élément repéré par une expression XPath
      - Imbriqués grâce à des balises d'application de templates

      ➔parallèle avec les fonctions en programmation déclarative

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : syntaxe
    - Élément racine

```
<xsl:stylesheet      version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

    - Éléments de premier niveau (cardinalité=0 ou 1)
      - `<xsl:output>` : définit le type d'arbre de sortie
        - Attribut `method` : 3 valeurs possibles (`text`, `html`, `xml`)
        - Autres attributs : `version`, `encoding`, `standalone`, `indent`…
      - `<xsl:include>` et `<xsl:import>` : permettent d'inclure d'autres feuilles de style
        - Attribut `href` : URI de la ressource à inclure
        - Différence entre les deux : règles de priorités

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : syntaxe
    - Éléments de premier niveau (cardinalité=0 ou 1)
      - `<xsl:strip-space>` et `<xsl:preserve-space>` : gestion des espaces dans l'arbre résultant (resp. suppression et conservation)
        - Attribut `elements` : noms des éléments concernés séparés par des espaces
      - `<xsl:template>` : modèle racine de l'arbre de sortie
        - Attribut `match` : désigne le nœud XPath concerné par l modèle (au premier niveau, toujours "/")
        - Contient la racine de la déclaration de l'arbre de sortie
      - Autres éléments (key, variable, attribute-set, param) : voi la recommandation

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les templates
    - Définition
      - Modèles simples : `<xsl:template match="noeud_XPath">`
        - L'expression XPath qui définit le nœud peut inclure un filtre
        - Ce nœud devient le nœud contextuel dans le template
      - Modèles nommés : `<xsl:template name="nom_tmplate">`
    - Appel
      - Modèles simples :

        `<xsl:apply-templates select="expr_XPath" />`
        - L'expression XPath est un chemin de localisation qui désigne le nœud
      - Modèles nommés :

        `<xsl:call-template name="nom_tmplate" />`

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Génération de contenus XML
      - `<xsl:element name="p" namespace="xhtml">Contenu de l'élément (ici: un paragraphe XHTML)</xsl:element>`

        Remarque : `<xsl:element>` n'est nécessaire que lorsque le nom de l'élément à générer doit être calculé
      - `<xsl:attribute name="href" namespace="xhtml">Contenu de l'attribut (ici : référence XHTML)</xsl:attribute>`

        Remarque : `<xsl:attribute>` se place dans l'élément auquel il se rapporte
      - `<xsl:text>Contenu textuel quelconque.</xsl:text>`

        Remarque : `<xsl:text>` ne sert qu'au formatage du texte (gestion des espaces…)
      - Tout autre élément XML bien formé est accepté

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Traitement de contenus de l'arbre XML source
      - `<xsl:value-of select="expr_XPath" />`
        - Permet d'obtenir la valeur d'un nœud (élément ou attribut)
        - L'expression XPath est un chemin de localisation
        - Elle désigne un nœud à partir du nœud contextuel
      - `<xsl:copy-of select="expr_XPath" />`
        - Permet de recopier dans l'arbre destination toute une partie de l'arbre source
        - L'expression XPath fonctionne comme précédemment
      - `<xsl:copy />`
        - Permet de copier uniquement un élément sans ses sous-éléments

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Structures de contrôle
      - `<xsl:if test="expr_XPath">Contenu conditionnel</xsl:if>`
        - Le contenu conditionnel peut être composé d'autres éléments (`<xsl:value-of select="expr_XPath" />`)
      - `<xsl:for-each select="expr_XPath">Contenu répété</xsl:for-each>`
        - Cet élément est redondant avec `<xsl:apply-templates />` mais rend la feuille de style moins lisibl

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Structures de contrôle
      - <xsl:choose>

```
<xsl:when test="expr_XPath1">
        Contenu conditionnel 1
</xsl:when>
<xsl:when test="expr_XPath2">
        Contenu conditionnel 2
</xsl:when>
...
<xsl:otherwise>
        Contenu conditionnel n
</xsl:otherwise>
```

</xsl:choose>

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Variables
      - <xsl:variable name="var" select="1" />
        ...
        <xsl:value-of select="$var" />
      - Attention, ce sont en fait des constantes
      - Portée : élément les contenant

Feuilles de style (X)HTML : CSS

- Buts : définir des caractéristiques de style
  - Standardisées pour être reconnues par tous les navigateurs
  - Dans un langage différent de HTML pour séparer la forme du fond
  - Externalisables pour pouvoir les réutiliser
  - Imbriquables pour pouvoir les appliquer à des sous-ensembles de documents

Feuilles de style (X)HTML : CSS

- Trois niveaux d'application
  - Niveau élément : feuilles de style intégrées
    - Syntaxe : caractéristiques de style dans la valeur de l'attribut `style` de la balise ouvrante
  - Niveau document : feuilles de style incorporées
    - Syntaxe : élément `<style type="text/css">` dans la balise <head> d'un document HTML
  - Niveau site web : feuilles de style liées
    - Syntaxe
      - Caractéristiques de style dans un fichier à part
      - Lien vers ce fichier par son URI dans un document HTML :

```
<link rel="stylesheet" type="text/css" href="URI" /
```

Feuilles de style (X)HTML : CSS

- Syntaxe du langage
  - Définition des caractéristiques de style

    ```
    nom_element {caract1:valeur1; caract2: valeur2;…}
    .nom_classe {caract1:valeur1; caract2: valeur2;…}
    nom_element.nom_classe {caract1:valeur1;…}
    ```

  - Application de styles
    - Aux éléments HTML : pas besoin de demander l'application
    - À des classes de style : application par l'attribut `class="nom_classe"`
  - Détail des caractéristiques : poly p. 37.

Outils de programmation avec XML

- Définitions
  - Qu’est-ce qu’un parser ?
    - « Un module logiciel [...] utilisé pour lire les documents XML et pour accéder à leur contenu et à leur structure. »
  - Qu’est-ce qu’une application ?
    - « On suppose qu'un processeur XML effectue son travail pour le compte d'un autre module, appelé l'application. »

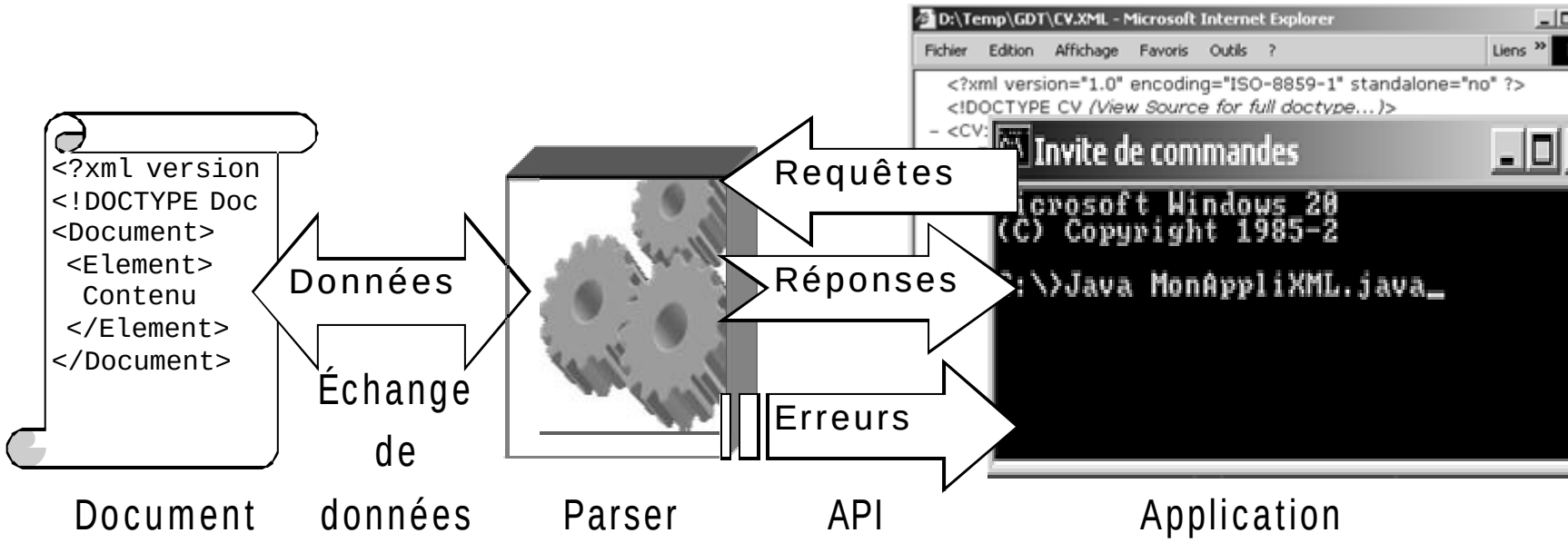
http://babel.alis.com/web_ml/xml/REC-xml.fr.html#dt-xml-proc

Partie 3

- Programmation avec XML
  - Principes généraux
  - Les API standard
    - DOM
    - SAX
  - XML et Java
  - XML et JavaScript

Programmation avec XML

- Communications entre parsers et applications
  - Rappel : Application Programming Interface
    - Outils
    - Protocole de communication
  - Schéma des échanges de données



Les API standard

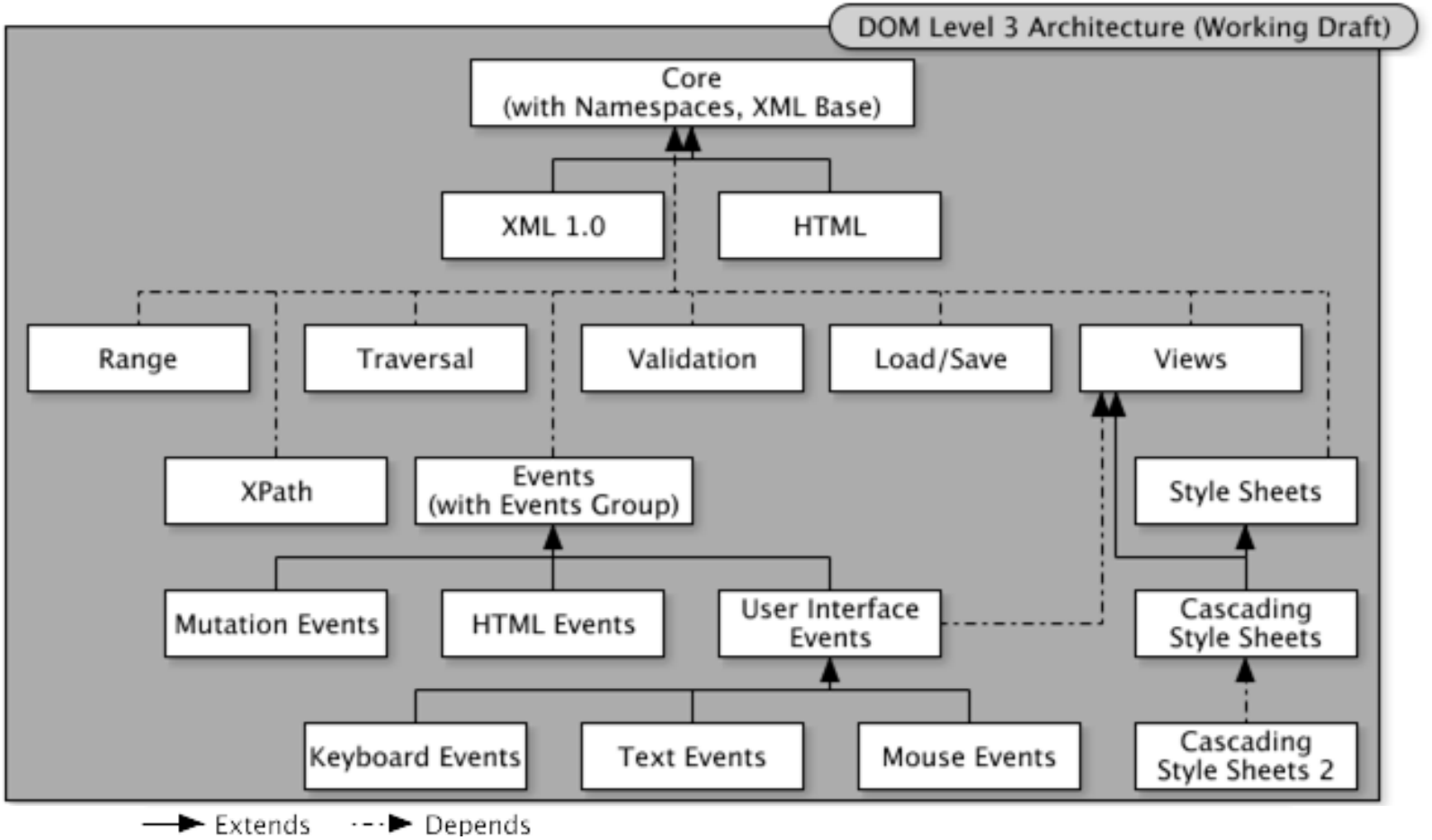
- Le DOM : généralités
  - Modèle objet de document
  - Motivations
    - Rendre les applications W3 dynamiques
    - Accéder aux documents HTML et XML depuis un langage de programmation
  - Utilisations courantes
    - Intégré aux navigateurs
    - Utilisé en programmation comme API XML
  - Origine : DOM working group (W3C)
    - Début : 1997 ; fin : ...
    - Standardiser les tentatives existantes

Les API standard

- Le DOM : principes fondamentaux
  - Représentation arborescente d'un document
    - Tout le document est chargé en mémoire
    - Navigation dans la structure arborescente
    - Représentation des nœuds par des *interfaces*
      - Propriétés
      - Méthodes
  - Recommandations sous forme de niveaux
    - Niveau 0 : avant…
    - Niveau 1 : octobre 1998
    - Niveau 2 : depuis novembre 2000
    - Niveau 3 : depuis janvier 2004

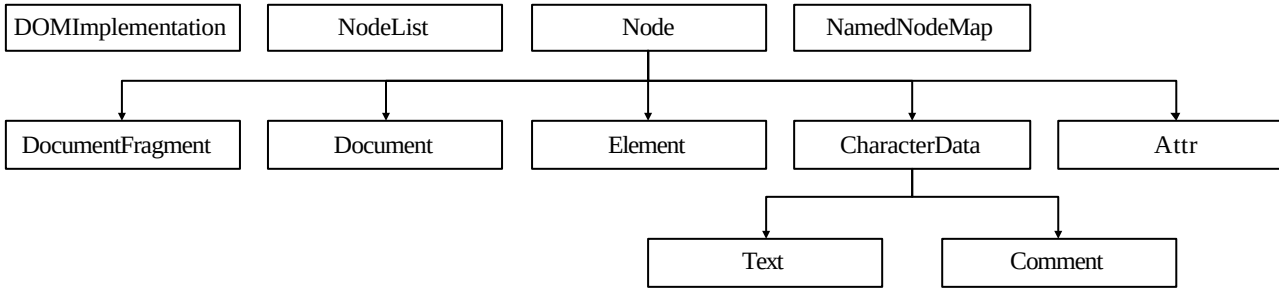
Les API standard

- Le DOM : fonctionnalités

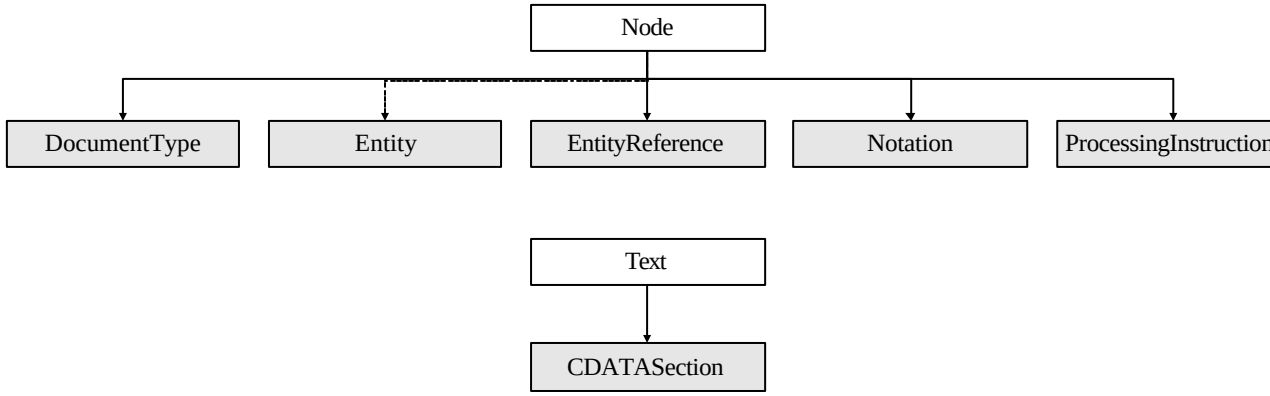


Les API standard

- Le DOM : modules
  - DOM Core :



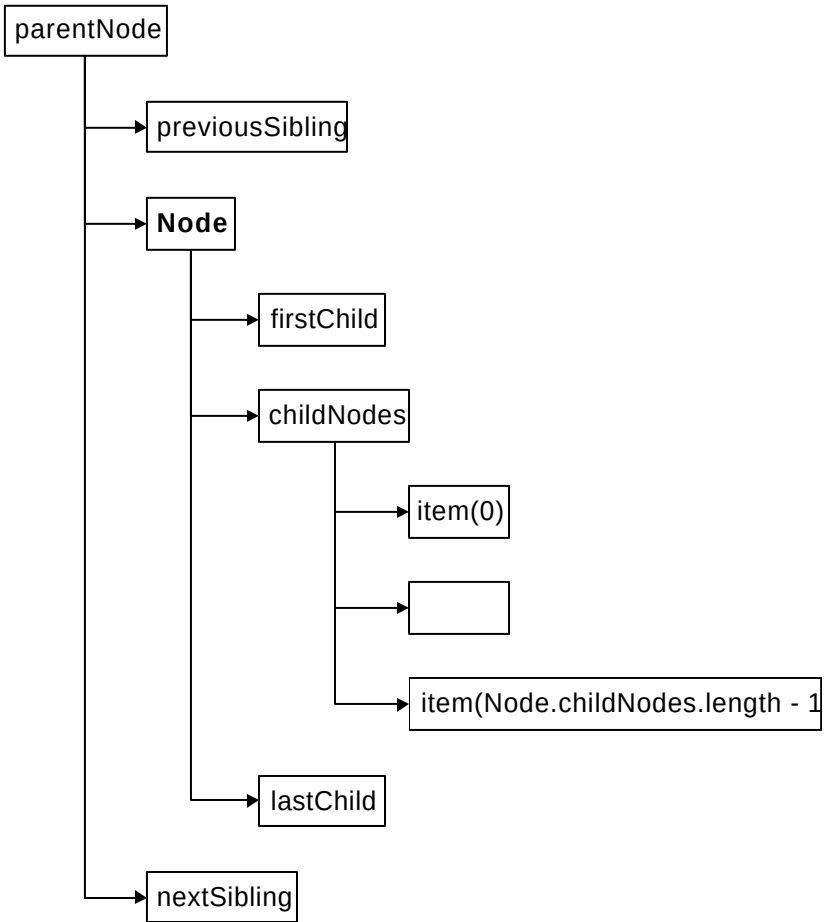
  - DOM XML :



  - Détail des interfaces : poly p. 93

Les API standard

- Le DOM : hiérarchisation des interfaces (module Core)



Les API standard

- Le DOM : utilisation en Java
  - Package JAXP `javax.xml.parsers`
  - Package spécifique `org.w3c.dom`
  - Liste des interfaces (Core + XML) : poly p. 113
  - Détail des interfaces : http://java.sun.com/j2se/1.4.2/docs/api

Les API standard

Exemple de code Java DOM

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import org.w3c.dom.Document;
import org.w3c.dom.DOMException;
public class DomParsing{
  static Document doc;
  public static void main()
  {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    try {
       DocumentBuilder builder = dbf.newDocumentBuilder();
       doc = builder.parse( new File("CV.xml") );
    ...}
    catch (ParserConfigurationException pce) { // Peut-être généré par la méthode
    }                                          // newDocumentBuilder()
    catch (SAXException se) {    // Peut être générée par la méthode parse()
    ...}
    catch (IOException ioe) {     // Peut être générée par la méthode parse()
    ...}
    catch (IllegalArgumentException iae) {  // Peut être générée par la méthode parse()
    ...}
  } // main
```

Les API standard

- SAX : principes fondamentaux
  - Simple API for XML
  - Issue d'une communauté de développeurs
    (liste xml-dev, sur http://www.xml.org)
  - Fondée sur la programmation événementielle
    - Pas de chargement de tout le document en mémoire
    - Pas de vision globale du document
  - À l'origine : développée en Java
  - Depuis : implémentations dans d'autres langages
  - 2 versions différentes

Les API standard

- SAX : principes fondamentaux
  - Des interfaces
    - Pour programmer des parsers compatibles SAX
    - Pour programmer des applications compatibles SAX
  - Des classes
    - Pour faciliter la programmation
    - Pour la gestion des erreurs
  - Des exceptions

Les API standard

- SAX : utilisation en Java
  - Procédure
    - Instanciation d'un parser
    - Lancement de l'analyse
    - Appel/implémentation de fonctions spécifiques
      - Interface `XMLReader` : `setDTDHandler()`
      - Interface `ContentHandler` : `startElement()`, `characters()`
      - Interface `Attributes` : `getLength()`, `getType`, `getValue()`
      - Interface `ErrorHandler` : `fatalError()`, `error()`, `warning()`

Les API standard

- SAX : utilisation en Java
  - Package JAXP `javax.xml.parsers`
  - Packages SAX `org.xml.sax`, `org.xml.sax.helpers`, `org.xml.sax.ext`
  - Présentation générale : poly p. 115
  - Détails

    http://java.sun.com/j2se/1.4.2/docs/api

Les API standard

Exemple de code Java SAX

```
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;

import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
public class SAXParsing extends DefaultHandler {
    public static void main()  {
        DefaultHandler dh = new SAXParser();
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
            SAXParser sp = factory.newSAXParser();
            sp.parse( new File("CV.xml"), dh);
        } catch (Throwable t) {...}
    }
    public void startDocument()  throws SAXException { ... }
    public void endDocument()   throws SAXException { ... }
 public void startElement(String namespaceURI, String LocalName, String QualifiedName,
Attributes atts) throws SAXException { ... }
    public void endElement(String namespaceURI, String LocalName, String QualifiedName)
throws SAXException { ... }
    public void characters(char buf[], int offset, int len) throws SAXException { ... }
    ...
}
```

XML et Java

- Standardisation des API
  - Nombreux parsers
  - API spécifiques
  - ⇒ Le DOM (W3C)
  - ⇒ SAX (xml-dev)
- Standardisation des accès aux parsers
  - Données conformes aux standards XML
  - Langage de programmation identique
  - Applications conformes aux API standards
  - Parsers implémentant ces API

  ?💣☠! Parsers différents pour faire la même chose

XML et Java

- JAXP : au départ
  - Java API for XML Parsing
  - Version 1.0

    Package Java additionnel au JDK 1.3
  - Couche intégration des parsers

    Instanciation du processeur transparente
  - Couche API

    Implémentation des API DOM et SAX

XML et Java

- JAXP : aujourd'hui
  - Java API for XML Processing
  - Version 1.2

    Package Java intégré au JDK 1.4
  - Couche intégration des parsers

    Instanciation du processeur transparente
  - Couche API
    - Implémentation des API DOM et SAX
    - Prise en charge des schémas XML
    - TrAX (Transformation API for XML)
      - XSLT
      - XSLTC

XML et Java

- JAXP : les packages java
  - Couche intégration des parsers

    `javax.xml.parsers`
  - Couche API
    - API DOM

      `org.w3c.dom`
    - API SAX

      `org.xml.sax`

      `org.xml.sax.helpers`

      `org.xml.sax.ext`
    - TrAX (Transformation API for XML)

      `javax.xml.transform`

XML et Java

- JDOM : l'alternative à JAXP ?
  - Représentation arborescente d'un document XML
  - Plus « facile » à utiliser que DOM et SAX
  - Compatible avec DOM et SAX
    - ⇒ Surcouche de DOM et SAX
  - Pas incompatible avec JAXP
  - Packages
    - `org.jdom ; org.jdom.adapter ; org.jdom.input ; org.jdom.output ; org.jdom.transform ; org.jdom.xpath`
  - Site web
    - http://www.jdom.org

XML et JavaScript

- Rappels sur le langage JavaScript
  - Scripting côté client (navigateurs web)
    - Intégré aux pages web

      ```
      <script language= "JavaScript"> function toto(){…} </script
      ```

    - Dans des fichiers séparés

      ```
      <script language= "JavaScript" src="fich.js"></script>
      ```

  - Caractéristiques du langage
    - Syntaxe proche de celle du C
    - Programmation fonctionnelle
    - Typage faible

```
function toto(){
  var texte = "toto";
  alert (texte);
}
```

XML et JavaScript

- Rappels sur le langage JavaScript
  - Modèles sous-jacents
    - Événementiel
      - Lié aux événements utilisateurs et de la page web
      - Association des scripts par des attributs HTML normalisés (`onload`, `onclick`, `onmouseover`…)
      - Utilisation des mécanismes de liens HTML
        `<a href="JavaScript:toto();">cliquez ici</a>`
    - Objet
      - Différents types d'objets natifs (String, Array, Object)
      - Interface DOM avec les documents XHTML et XML
      - Implémentations spécifiques aux moteurs
        - Gecko (Mozilla, Safari) : ECMAScript
        - MicroSoft (IE) : Jscript

      ➔ Nécessite des tests d'existence des interfaces et méthode

Asynchronous Javascript And XML (AJAX)

- Principe
  - Applications web avec interface utilisateur
  - Déporter un maximum de code sur le client
    - Réduction des ressources consommées côté serveur
    - Réduction de la bande passante réseau
- Exemples d'utilisation
  - Google (mail, earth…)
  - Suggestions automatiques
  - Traitement de texte
- Frameworks
  - openAjax (IBM) : Dojo, Rico ; Ruby / Ruby on Rails (RoR)
  - Plugin Eclipse : Rich Ajax Platform

Asynchronous Javascript And XML (AJAX)

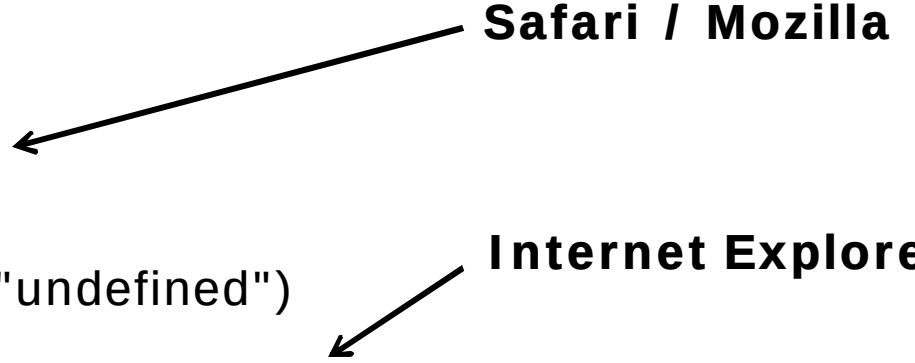
- Fonctionnement
  - Requête asynchrone au serveur dans une fonction javascript (déclenchée par un événement quelconque)
  - Transfert asynchrone de données en XML
  - Chargement d'un document XML (API DOM) dans un
    - Objet XMLHttpRequest (Mozilla)
    - Contrôle ActiveX XMLHTTP (IE)
  - Traitement dynamique côté client
    - Affichage (transformation XSLT)
    - Logique applicative (fonctions JavaScript dédiées)

Asynchronous Javascript And XML (AJAX)

- Exemple de code : création d'un objet requête

```
var req = null;

function getRequest()
{
   if (window.XMLHttpRequest)
   {
      req = new XMLHttpRequest();
   }
   else if (typeof ActiveXObject != "undefined")
   {
      req=new ActiveXObject("Microsoft.XMLHTTP");
   }
   return req;
}
```



Asynchronous Javascript And XML (AJAX)

- Exemple de code : chargement asynchrone

```
function GetDataUsingAJAX (HttpMethod, url, params, elt)
{
   if(req != null)
   {
      // méthode avec paramètres
      req.onreadystatechange = function() {stateChange(elt)};
      // méthode sans paramètre
      // req.onreadystatechange = stateChange;

      req.open(HttpMethod, url, true);
      req.setRequestHeader("Accept", "application/xml");
      req.send(params);
    }
}
```

Assiciation de la fonction de gestion de l'état

Asynchronous Javascript And XML (AJAX)

- Exemple de code : gestion de l'état

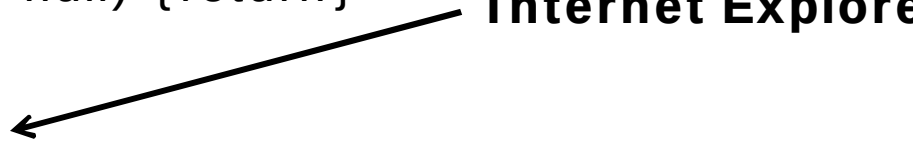
```
function stateChange (elt)
{
   if(req.readyState == 4) {
      if (req.responseXML != null) {
         var docXML= req.responseXML;
      } else {
         var docXML= req.responseText;
         docXML=parseFromString(docXML);
      }
      var docXMLresult = traiteXML(docXML);
      var str = (new XMLSerializer()).serializeToString(docXMLresult);
      document.getElementById(elt).innerHTML += str;
   }
}
```

READY_STATE_COMPLETE

Asynchronous Javascript And XML (AJAX)

- Exemple de code : transformation XSLT

```
//Après chargement asynchrone des documents XML et XSLT
function transform XSLT (XMLDoc, XSLDoc, id)
{
   if(XMLDoc == null || XSLDoc == null) {return}
   try {
      if (window.ActiveXObject)
      {
         var target = document.getElementById(id);
         target.innerHTML = xml.transformNode(xsl);
      }
```



Asynchronous Javascript And XML (AJAX)

- Exemple de code : transformation XSLT

```
    } else if (window.XSLTProcessor) {
        var fragment;
        var xsltProcessor = new XSLTProcessor();
        xsltProcessor.importStylesheet(xsl);
        fragment = xsltProcessor.transformToFragment(xml, document);
        var target = document.getElementById(id);

        target.appendChild(fragment);
        document.appendChild(target);
    }
  } catch (e) {
    return e;
  }
}
```

Safari / Mozilla

Asynchronous Javascript And XML (AJAX)

- implémentation de la logique applicative
  - Programmation directe en JavaScript
    - Fonctionnalités limitées
    - Fait parfois un peu « fouillis »
  - Utilisation de frameworks
    - Programmation dans un autre langage
    - Génération du code JavaScript

XML et JavaScript

- Exemple
  http://www710.univ-lyon1.fr/~lmedini/Miage-soir/Exemple.html
- Références
  - JavaScript en général :
    http://fr.wikipedia.org/wiki/JavaScript
  - ECMAScript :
    http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf
  - Jscript :
    http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/29f83a2c-48c5-49e2-9ae0-7371d2cda2ff.asp
  - AJAX :
    http://www.xul.fr/xml-ajax.html
    http://developer.mozilla.org/fr/docs/AJAX