# Document et Web Sémantique - TP Validation à l'aide de schémas XSD

L'objectif de ce TP est d'utiliser le CI/CD de gitlab pour la validités des fichiers XML à l'aide de schémas XSD.

## Préambule

Vous utiliserez `xmllint` pour valider un document XML à l'aide d'un schéma XSD. Il faudra utiliser les options suivantes :

**`noent`** pour interpréter les entités ;

**`noout`** pour éviter de produire le fichier XML sur la sortie standard lorsque celui ci est valide ;

**`schema`** pour indiquer que l'on veut une validation à l'aide d'un schéma XSD.

Le schéma global est décomposé en plusieurs schémas, chacun validant un type de fichier XML.

**Question :** pourquoi XSD propose deux instructions, `xsd:import` et `xsd:include`, pour inclure un schéma dans un autre ?

Les schémas sont dans le répertoire `xsd` :

**`specialite.xsd`** pour valider le fichier XML général, décrivant une spécialité, comme par exemple `iti.xml` ;

**`contrat.xsd`** pour valider le fichier XML décrivant un contrat d'un semestre, ceux se trouvant dans le répertoire `contrats` ;

**`ec.xsd`** pour valider le fichier XML décrivant un EC, les fichiers se trouvent dans le répertoire `ecs` ;

**`enseignants.xsd`** pour valider le fichier XML décrivant les enseignants : `enseignants.xml` ;

**`competences.xsd`** pour valider le fichier XML décrivant les savoir-agir, jalons et apprentissages critiques : `competences.xml`.

Il y a aussi de schémas décrivant des types simples et types complexes utilisés dans les précédents schémas :

**`texte.xsd`** proposant des types liés au texte (extrait du HTML, `LangString`, etc.)

**`general.xsd`** proposant les types `TDuree` et `TDecimalPositif`.

Enfin le schéma `xml.xsd`, issu du w3c, décrit entre autre le type simple `xml:lang`.

A chaque fois que vous choisissez un type XSD, sélectionnez le type le plus contraint. Pour rappel, pour les chaînes de caractères, XSD propose les types :

**`string`** Permet d'utiliser n'importe quel texte, y compris plusieurs espaces consécutifs, tabulations et retours à la ligne. Il conserve les espaces multiples.

**`normalizedString`** Supprime les retours à la ligne et tabulations, et remplace les espaces multiples par un seul. Il ne supprime pas les espaces en début ou fin de texte.

**`token`** Supprime les retours à la ligne, les tabulations et les espaces en début et fin de texte. Réduit les espaces multiples à un seul. Il conserve les caractères spéciaux.

**`NMTOKEN`** Permet uniquement des identifiants composés de lettres, chiffres et certains caractères spéciaux (`-`, `_`, `.`, `:`). Il n'accepte aucun espace ni caractère interdit dans un nom XML.

# 1 `enseignants.xsd`

Complétez le fichier `enseignants.xsd` tel qu'il définit :

**`TDepartement`** un énuméré avec les valeurs `ITI`, `GM`, `HUMA`, `STPI`, `EXT`, `DSI`, `MECA` ;

**`TEnseignant`** un type complexe décrivant un enseignant ;

**`TEnseignants`** un type complexe décrivant une liste d'enseignants ;

**enseignants** un élément racine de type `TEnseignants`.

Vérifiez la validité du fichier XML `enseignants.xml`.

# 2 contrat.xsd

Complétez le fichier `contrat.xsd` tel qu'il définit :

**`TChoix`** un énuméré avec les valeurs `Obligatoire`, `Electif` et `Facultatif` ;

**`TECDansUE`** un type complexe qui décrit les éléments EC au sein des élément UE.

**`TUE`** un type complexe qui décrit une UE ;

**`TContrat`** un type complexe qui décrit un contrat ;

**`Contrat`** un élément racine de type `TContrat`.

Vérifiez les validités des fichiers XML se trouvant dans le répertoire `contrats`.

**Questions** :
— Pourquoi n'avons nous pas appeler le type `TECDansUE` tout simplement `TEC` ?
— Quelle limite au pouvoir d'expression de XSD avez vous identifié ?

# 3 Ajout de la vérification au niveau du CI/CD

Ajoutez un nouveau *stage* et un nouveau *job* pour vérifier le fichier `iti.xml`.