Chapitre V

TP - Premier contact avec UML, XML et XML Schema

1 Démarrage

Au début de ce module, il est possible que certains ne disposent pas encore d'une adresse universitaire (etu.univ-grenoble-alpes). Dans ce cas, vous pourrez réaliser les premiers TPs sous **Netbeans** qui est disponible sur les machines de l'université.

Sinon, vous utiliserez **Jetbrains** (idéalement **Rider**, sinon **IntelliJ**). Ce logiciel est payant normalement, mais gratuit pour qui dispose d'une adresse universitaire.
JetBrains est installé sur les machines de la fac. Vous pouvez aussi décider de l'installer sur votre machine personnelle (dans ce cas, installez **Jetbrains Rider**).
Dans les deux cas, pour pouvoir l'utiliser, vous devez disposer d'une licence.
Si vous disposez d'une adresse universitaire grenobloise valide, vous devez vous rendre à cette adresse https://www.jetbrains.com/community/education/students/ et cliquer sur **Apply now** dans le bandeau violet titré **Get free access to all developer tools from JetBrains!**. Suivez les instructions et récupérez votre numéro de licence. Vous pouvez ensuite l'utiliser pour l'usage de Jetbrains.

2 Création d'un projet JetBrains Rider / Netbeans

Si vous pouvez utiliser Jetbrains, vous devez un projet C#. Sinon vous pouvez pour commencer, générer un projet Java sous Netbeans. Ce projet Java sera ensuite rapidement remplacé pr un projet C# sous Jetbrains Rider.

2.a Jetbrains Rider

Creez un nouveau projet de type `dotnet console` : depuis le menu : `New solution`. Incluez la gestion de git (création de repository git) ; cela vous permettra de sauvegarder l'avancée de votre projet. Cf la figure V.1.

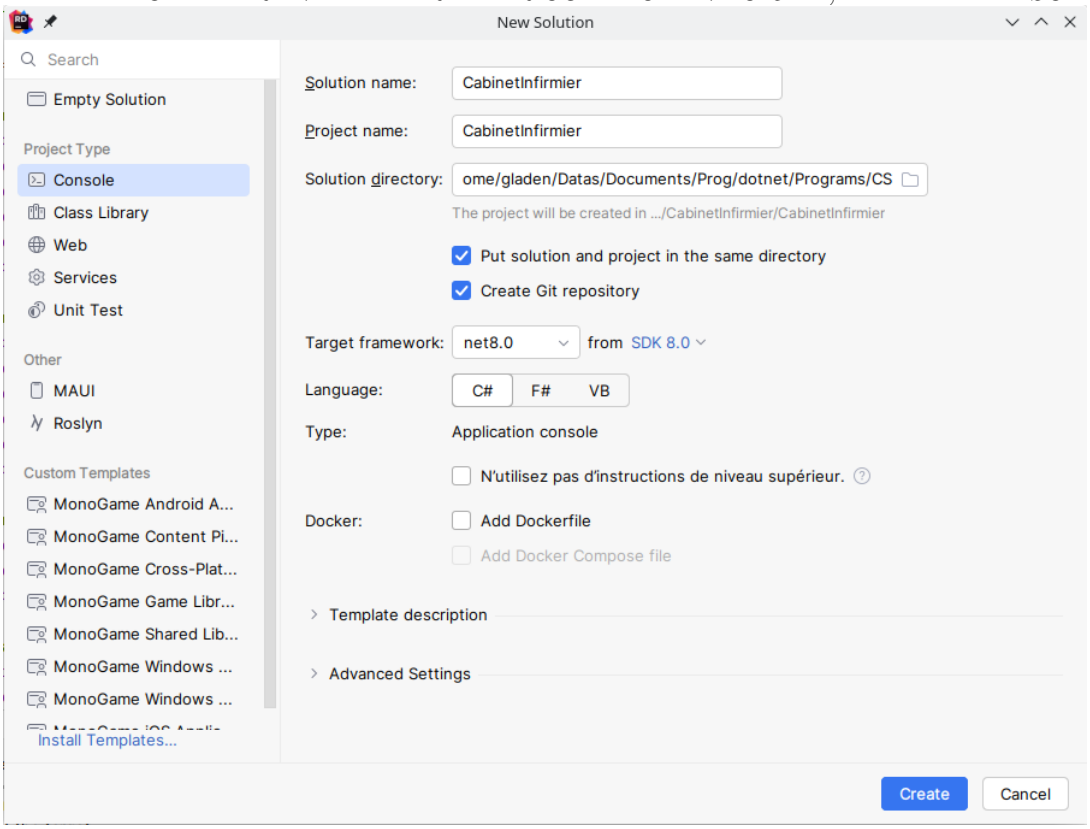

Figure V.1: **Création du projet (solution) Cabinet Infirmier sous Jetbrains Rider).**

Dans ce projet, dans le dossier `src`, vous créerez l'arborescence montrée figure V.2.

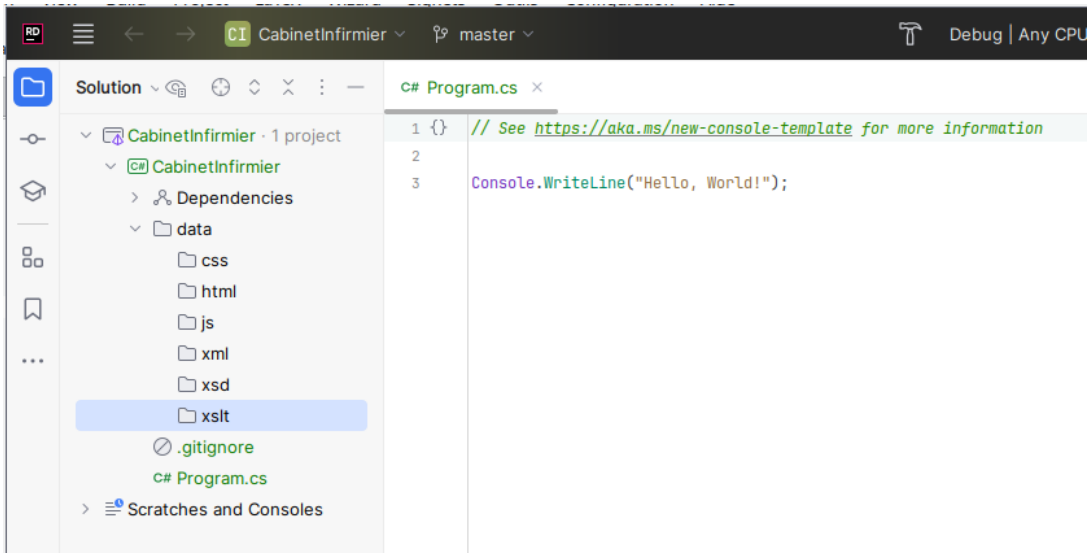

Figure V.2: **Arborescence des données dans le projet Cabinet Infirmier.**

2.b Netbeans (solution de secours en début d'année)

Pour chaque Exercice, créez un projet Java de type `Java with Ant`. Nommez le `Exo1`, `Exo2` ...

Pour le projet `Cabinet Infirmier`, créez un projet Java de type `Java with Ant` que vous nommerez `CabinetInfirmier`. Dans ce projet, dans le dossier `src`, vous créerez l'arborescence montrée figure V.3.

NB : dès que vous pourrez passer sous Jetbrains, vous créerezun projet C# dans lequel vous copierez l'ensemble de cette arborescence.

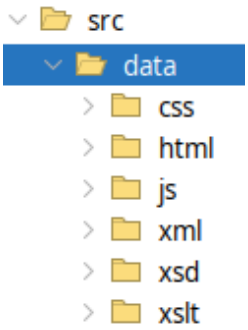

Figure V.3: **Arborescence des données dans le projet Cabinet Infirmier**.

3 Exercices de prise en main

Ces exercices ne sont pas rendus et ne font pas l'objet de notation. Ils sont cependant votre assurance de comprendre le cours en vous exerçant. Il est indispensable de les faire avant de s'attaquer à un quelconque projet.

3.a Un premier document XML ... et son diagramme UML (Exo1)

*** 1ère étape.**

Créez un nouveau document XML nommé `planning.xml`. Testez l'usage des outils de l'IDE (voir chapitre II) pour :

- vérifier la conformité XML
- vérifier la validité par rapport à un schéma (votre document XML de base n'étant pas contraint par un schéma

*** 2ème étape.**

Considérons le document suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<week>
  <date>20 oct</date>
  <tutorial>Algorithmique : statistiques</tutorial>
  <lecture>Boucles et conditions en Java
  </Lecture>
</week>

<holidays>
  <date>27 oct</date>
</holidays>

<week>
  <date>3 nov</date>
  <tutorial>Algorithmique : tableaux</tutorial>
  <Lecture>Les tableaux N dimensions
  </lecture>
</week>
```

Remplacez le contenu du document `planning.xml` par celui-ci. Le document est il bien formé ? Pourquoi ? Prenez les mesures qu'il faut pour corriger le document si nécessaire.

*** 3ème étape.**

Réalisez un diagramme UML de ce document, sur papier d'abord, sur PlantUML ensuite.

Papier. Réalisez scrupuleusement toutes ces étapes :

- Commencez par identifier à partir du document le ou les types dont vous avez besoin : listez les.
- Identifiez quels types seront des restrictions (des types simples restreints) et comment.
- Dessinez sur papier un schéma UML contenant tous les types et leurs relations (composition, ...)
- N'oubliez pas d'englober ces types dans un namespace. Le nom du vocabulaire sera `http://www.univ-grenoble-alp`

UML - PlantUML Vous allez maintenant implémenter votre diagramme dans le langage de script `PlantUML`. Vous trouverez toute la documentation nécessaire sur ce site, en particulier ici : documentation pour les diagrammes de classe.

Pour tester vos scripts et générer vos diagrammes sous forme d'images, vous devez les soummettre dans le service en ligne PlantUML.

Progressez doucement, par petites implémentations.

3.b Un premier Schema XML, son diagramme UML ... et son instance XML (Exo 2)

Cette fois, vous partez d'un schéma XML. A partir de ce schéma que vous corrigerez si nécessaire, vous concevrez un diagramme UML (papier d'abord, PlantUML ensuite), puis génèrerez une instance de document XML que vous complèterez (données valides par rapport au schéma) ; comme vous avez des profs sous la main, trouver leurs noms, prénoms et spécialités pour la filière Miage ne devrait pas poser de problèmes.

Voici ce Schema XML :

```
<?xml version="1.0"?>
<xs:schema xmlns:ens="http://www.univ-grenoble-alpes.fr/enseignants"
        xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">

    <!-- Element racine -->
    <element name="enseignants" type="Enseignants"/>

    <!-- Le type Enseignants contient une séquence d'"Enseignant"
        ainsi que la filière et l'année dans laquelle ils enseignent.
    -->
    <complexType name="Enseignants">
        <attribute name="filiere" type="string"/>
        <attribute name="annee" type="gYear"/>
        <sequence>
            <element name="enseignant" type="ens:Enseignant" maxOccurs="
                unbounded"/>
        </sequence>
    </complexType>

    <!-- Le type Enseignant contient un nom, un prénom et une spécialité
        Il est identifié par une référence.
     -->
    <complexType name="Enseignant">
        <sequence>
            <element name="nom" type="ens:Denomination"/>
            <element name="prenom" type="ens:Denomination"/>
```

```
            <element name="specialite" type="ens:Specialite"/>
        </sequence>
        <attribute name="ref" type="xs:string"/>
    </complexType>

    <!-- Le type Denomination contraint les noms et prénoms -->
    <simpleType name="Dénomination">
        <restriction base="string">
            <pattern="[A-Z][a-z]*"/>
        </restriction>
    </simpleType>

    <!-- Le type Specialite contraint le choix des spécialités -->
    <simpleType name="Specialite">
        <restriction base="string">
            <enumeration type="APO"/>
            <enumeration type="FDD-XML"/>
            <enumeration type="IHM"/>
            <enumeration type="RO"/>
        </restriction>
    </simpleType>

</xs:schema>
```

Question subsidiaire : quelles sont les pistes d'amélioration pour un tel schéma ? Comment procèderiez vous ?

3.c Un premier UML ... et tout le reste (Exo 3)

Cette fois, nous allons partir d'un énoncé. Ce sera à vous d'imaginer un diagramme UML (papier + PlantUML) permettant de modéliser cette situation. A partir de ce que vous aurez proposé comme diagramme, créez le schéma XML complet correspondant en fixant un namespace, puis générez un fichier XML que vous renseignerez.

Voici l'énoncé :
On veut concevoir un jeu vidéo se passant dans une salle de classe. Le jeu consiste pour un personnage, le professeur, à répondre aux questions d'étudiants en TP de FDD-XML atablés chacun devant un ordinateur. Chaque salle de classe représente un niveau de jeu différent. On souhaite ici modéliser un niveau de jeu.
Un niveau de jeu tel qu'on souhaite le modéliser contient :

- un temps de jeu limite de type entier
- la structure d'une salle de classe

Une salle de classe est constituée d'un nombre compris entre 2 et 20 rangées (ou lignes) de tables.
Une ligne de tables est modélisée par une chaine de caractère contenant un nombre indéterminé de valeurs `0`, `1` ou `2`.

4 Projet Cabinet Infirmier

- Lisez **très attentivement** la description du projet infirmier (chapitre III
- Durant les 4 séances à venir, vous réaliserez la partie modélisation (section 4 du chapitre III) du projet Cabinet Infirmier. Attention, ce travail (partie XML du cabinet infirmier) sera récupéré vers la moitié du module et noté. Vous devez donc gérer votre temps en évaluant combien de temps vous passez sur chaque réalisation.