



Cours Modélisation XML et sérialisation des données

Resp : Anis jedidi

Auditoires: P-ING GENIE INFORMATIQUE

Partie : Sérialisation

09/12/2025

1

1

PLAN

- C'est quoi la sérialisation?
- C'est quoi la désérialisation?
- Quelles sont les différents types de sérialisation?
- C'est quoi la sérialisation et la désérialisation XML?

09/12/2025

2

2

Le Problème

- **Scénario : Jeu Vidéo**

Vous jouez à un jeu. Vous êtes au niveau 10, avec un inventaire complexe, une position en X,Y,Z...

- **Comment [] cet état complexe ?**

L'objet jeu en mémoire RAM est vivant, mais :

- ✗ RAM se vide quand on éteint l'ordinateur
- ✗ Impossible d'envoyer la mémoire brute par réseau
- ✗ Impossible de le partager avec un programme Python

09/12/2025

3

3

L'Analogie du Déménagement

- **Le Meuble Monté (Java Object)**

Vous avez acheté une bibliothèque « NOBLE » chez « MEUBLATX ».

Elle est **montée** dans votre salon.

- ✓ Pratique à utiliser (appeler ses méthodes)
- ✓ Belle en 3D
- ✓ Occupe l'espace

Vous voulez la déplacer ...??

- ✗ Impossible à faire passer par la porte
- ✗ Impossible à envoyer par transporteur « DHL »

09/12/2025

4

4

L'Analogie du Déménagement

- **Le Carton (Fichier XML)**

Pour déménager, vous **démontez** la bibliothèque.

- ✓ Pièces à plat
- ✓ Format standard (carton)
- ✓ Passe par la porte
- ✓ Transportable par DHL
- ✓ Lisible par n'importe qui (notice universelle)

09/12/2025

5

5

Le Processus Complet

①

Démontage

Meuble 3D → Pièces plates + Instructions

②

Déménagement

Carton par la porte → Camion → Destination

③

Remontage

Pièces + Instructions → Meuble 3D

09/12/2025

6

6

Les Acteurs de cette Analogie

= Meuble monté dans le salon

= Démontage en pièces

= Carton avec pièces + notice

= Remontage du meuble

= Notice MEUBLATEX (comment monter)

09/12/2025

7

7

Sérialisation ? Désérialisation?

- La sérialisation c'est la conversion des propriétés et champs publics d'un objet en format XML.
 - Utilisée pour le stockage ou le transport des données.
 - La désérialisation c'est la recréation de l'objet d'origine à partir du XML.

09/12/2025

8

8

La sérialisation ?

- Sérialisation est le processus de conversion de l'état d'un objet en **une forme transportée à travers les processus/machines**
- L'objet est sérialisé dans un flux qui porte non seulement les données, mais aussi les informations sur le type de l'objet, telles que sa version, sa description et son nom d'assemblage,
 - A partir de ce flux, l'objet peut être trié dans une base de données, un fichier ou une mémoire

09/12/2025

9

9

La sérialisation ?

- La sérialisation correspond au processus de conversion d'un objet en **un formulaire universel facilement transportable : XML.**
 - Par exemple : sérialiser un objet et le transporter par Internet via HTTP entre un client et un serveur
- **Avantages :**
 - **Interopérabilité** : Échange de données entre différentes plateformes.
 - **Lisibilité** : Fichiers XML lisibles par les humains.
 - **Flexibilité** : Représentation de structures de données complexes.

09/12/2025

10

10

La sérialisation ?

La sérialisation est un concept fondamental dans le développement logiciel moderne. ➔ Elle permet de convertir des objets en un format qui peut être facilement stocké ou transmis et ensuite reconvertis en objets.

Ceci garantit :

- **Interopérabilité** : XML est un format de données universel, ce qui facilite l'échange d'informations entre différentes plateformes et langages de programmation.
- **Stockage Persistant** : Les données sérialisées en XML peuvent être stockées dans des fichiers ou des bases de données, permettant une récupération facile et fiable.
- **Communication Réseau** : XML est souvent utilisé pour transmettre des données sur le réseau, notamment dans les services web et les API
- **Facilité de Lecture et de Débogage** : Contrairement aux formats binaires, XML est lisible par les humains, ce qui simplifie le débogage et la maintenance des applications.

09/12/2025

11

11

La sérialisation ?

- La sérialisation est un procédé qui consiste à sauvegarder l'état (ou les données) d'un objet dans un fichier plutôt que de le garder en mémoire.
 - Ce mécanisme nécessite dès que les objets doivent être accessibles à partir d'un environnement étranger (services web , par exemple) : un objet peut ainsi exister entre deux programmes ou deux exécutions d'un programme (c'est la persistance objet)

09/12/2025

12

12

Finalité 1 : La Persistance

- **Scénario : Jeu Vidéo**

- Sauvegarder l'état de votre application et la restaurer plus tard. : Vous jouez à un jeu, vous quittez la partie, mais vous voulez garder l'état de votre personnage sauvegardé (niveau, inventaire) Vous êtes au niveau 10, avec un inventaire complexe, une position en X,Y,Z... :
 - Joueur sauvegarde à niveau 10 → Objet GameState sérialisé en XML
 - Fichier game_save.xml écrit sur disque
 - Joueur redémarre → Fichier XML lu et déserialisé
 - Joueur reprend au niveau 10 ✓

09/12/2025

13

13

Application de gestion des utilisateurs

Un utilisateur remplit son profil (nom, âge, email). On sauvegarde cet état dans un fichier XML pour qu'il puisse être récupéré plus tard, même si le programme est fermé.

- **Étape 1 : Définir la classe User**
 - On définit la classe User avec les attributs que l'on veut sérialiser
- **Avant sérialisation :**
 - L'objet user existe uniquement en mémoire. Si le programme se ferme, toutes ses données sont perdues.
- **Étape 2 : Sérialiser l'objet en XML (persistance)**
 - On transforme l'objet en XML et on le sauvegarde dans un fichier (user.xml).
 - Résultat après la sérialisation : Le fichier user.xml
- **Après sérialisation :**
 - L'objet est transformé en XML et sauvegardé dans un fichier (user.xml). Ce fichier peut être lu par un autre programme ou restauré plus tard.
- **Étape 3 : Désérialiser l'objet à partir du fichier XML**
 - On lit le fichier user.xml pour restaurer l'objet.
 - Sortie console après désérialisation : yaml (selon le format souhaité...)
- **Après désérialisation :**
 - Le fichier XML est lu pour recréer l'objet User en mémoire avec ses données originales (nom, âge, email).

09/12/2025

14

14

Sérialisation XML et persistance objet

- La **persistance objet** signifie que l'état d'un objet (ses données) peut être sauvegardé dans un format durable (comme un fichier XML) et rechargé plus tard pour reprendre son état.
- **Les exemples étudiés**
 1. **Jeux vidéo** : Un joueur quitte une partie, mais l'état de son personnage est sauvegardé (niveau, inventaire) dans un fichier XML.
 2. **Sauvegarde des sessions utilisateur** : Une application web peut sauvegarder les informations d'un utilisateur pour qu'il puisse reprendre sa session plus tard.
 3. **Applications de gestion** : Une application sauvegarde et restaure des données comme les commandes ou les clients.

09/12/2025

15

15

Finalité 2 : L'Échange

- Partager des données avec d'autres applications ou langages.
- **Exemple : API Web**
 - Application Java crée un objet "Produit"
 - Sérialise en XML → API REST
 - Client Python reçoit le XML
 - Python peut déserialiser et utiliser les données ✓
 - Aucune dépendance Java nécessaire !

09/12/2025

16

16

En Une Phrase

- La sérialisation transforme un objet vivant (en mémoire) en un format texte universel (XML) pour le sauvegarder ou le partager.

Objet Java ↔ Fichier XML

Pas de danger, pas de menace pour de données
si on respecte la "notice"
(Schéma XSD)

Au moment de la Désérialisation

09/12/2025

17

17

La Désérialisation?

- L'opération inverse de la sérialisation, est celle qui consiste à récupérer ces objets, s'appelle la désérialisation
- C'est le processus qui **Reconvertit** le résultat de la sérialisation dans l'état de l'objet original

**La sérialisation/désérialisation consiste à convertir tout (ou une partie) des valeurs des attributs d'une classe.
Le code des méthodes ou des propriétés n'est pas sérialisé.**

09/12/2025

18

18

Le Défi Technique

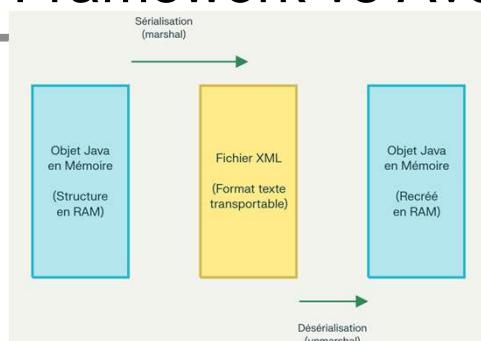
- **Comment automatiser ce processus ?**
 - Écrire du code pour démontage/remontage à chaque fois c'est compliqué.
- Solution : Les Framework de binding (plusieurs plateformes : c#, .net, microsoft, API-java, ...)
- **Nous utiliserons : JAXB (Jakarta XML Binding)**
 - JAXB = "Moyen automatique de convertir Java ↔ XML"
 - Il fait le démontage/remontage à notre place

09/12/2025

19

19

Sans Framework vs Avec JAXB



✗ Vous écrivez le code

- Lire chaque champ
- Vérifier le type
- Créer des balises
- 100+ lignes
- Erreurs faciles**

✓ JAXB le fait

- Annotations simples
- Une seule fonction
- Validation auto
- 10 lignes
- Fiable**

09/12/2025

20

20

Sérialisation - Désérialisation

- Pour sérialiser (ou désérialiser) une classe, deux étapes sont nécessaires
 - Spécifier explicitement dans la classe, les champs (ou les valeurs des propriétés) que l'on souhaite sérialiser.
 - Utiliser un sérialiseur : c'est cette classe qui permet de sérialiser ou de désérialiser des instances de la classe précédemment modifiée.
 - Un sérialiseur peut sérialiser ou désérialiser des objets au format XML (aussi binaire c'est possible) : intégré dans des Frameworks et éditeurs tiers...

09/12/2025

21

21

Différents types de sérialisation

- Formats lisibles à la XML :
 - XML
 - Lisible → Java.beans
 - Lisible → C# .net
 - Lisible → API SAX : JAXB
 - JSON (essentiellement lié à JavaScript)
 - XRD (External Data Representation)
 -

09/12/2025

22

22

Les Approches (API XML)

- DOM, SAX, JAXB : Quelle API pour quel besoin ?
- **Les Trois Niveaux d'Abstraction**
 - BAS NIVEAU (Bas niveau d'abstraction)**
 - DOM, SAX : Manipulation fine, contrôle total sur le XML
→ Compliqué
 - NIVEAU MOYEN**
 - StAX : Streaming, bon compromis → Pour flux volumineux
 - HAUT NIVEAU (Abstraction maximale)**
 - JAXB : Mapping automatique Java ↔ XML → Simple et efficace

09/12/2025

23

23

DOM (Document Object Model)

- **Analogie : Bricoler planche par planche**
 - DOM charge le **document entier en mémoire** sous forme d'arbre.
- **Vous pouvez :**
 - Parcourir l'arbre (parents, enfants, frères)
 - Ajouter/supprimer/modifier des nœuds
 - Accès aléatoire (n'importe quel nœud, n'importe quand)
 - Contrôle total = Code très verbeux
- **Pour la sérialisation** : Trop de codes, peu utilisé pour mapper des objets

09/12/2025

24

24

SAX (Simple API for XML)

- **Analogie : Lire la notice instruction par instruction**
 - SAX lit le fichier séquentiellement et lance des événements.
- **Caractéristiques :**
 - Événements : "début élément", "fin élément", "texte"
 - Pas de document en mémoire = Très léger
 - Parfait pour fichiers XML énormes
 - Accès séquentiel uniquement
- **Pour la sérialisation :** Peu adapté au mapping objet-XML

09/12/2025

25

25

JAXB (Jakarta XML Binding)

- **Analogie : La notice de Meuble automatisée**
 - JAXB transforme automatiquement objets Java ↔ XML via des annotations.
 - **Caractéristiques :**
 - Annotations simples (`@XmlRootElement`, `@XmlElement`)
 - Conversion automatique (`marshal / unmarshal`)
 - Validation contre schéma XSD
 - Peu de code = Pas d'erreur
- IDÉAL pour la sérialisation**: C'est exactement ce qu'on a besoin !

09/12/2025

26

26

Besoins ... API XML

- **Si l'objectif est la sérialisation / désérialisation d'objets Java en XML :**
 - **JAXB** (idéal pour mapper directement des objets et des fichiers XML).
- **Si l'objectif est le contrôle total sur la structure XML (sans être lié à des objets Java) :**
 - **DOM ou JDOM** pour construire et manipuler des documents XML.
- **Si l'objectif est de lire des fichiers XML volumineux / complexes, ou générer du XML efficacement :**
 - **SAX ou StAX**, mais ils ne sont pas conçus pour la sérialisation d'objets.

09/12/2025

27

27

API	Appropriée pour sérialisation ?	Comment ?	Avantages	Inconvénients
JAXB	✓ Oui	Sérialisation/Désérialisation automatique basée sur des annotations	Simple, rapide, adapté aux objets Java complexes	Moins flexible pour des structures XML complexes
SAX	✗ Non	Nécessite un code personnalisé pour écouter les événements SAX	Léger, performant, ne consomme pas beaucoup de mémoire	Nécessite beaucoup de code manuel pour la sérialisation
DOM	⚠ Possible	Construire manuellement un arbre DOM et sauvegarder	Flexible pour créer ou manipuler des fichiers XML complexes	Consomme beaucoup de mémoire pour des fichiers XML volumineux
JDOM	⚠ Possible	Construire un document JDOM à partir d'objets Java	Plus simple que DOM pour manipuler des fichiers XML	Nécessite une implémentation manuelle pour convertir les objets Java
StAX	✗ Non	Lecture/écriture progressive de fichiers XML	Performant pour les flux XML volumineux	Pas conçu pour mapper directement des objets Java en XML
XStream	✓ Oui	Sérialisation/Désérialisation automatique comme JAXB	Supporte plusieurs formats (XML, JSON)	Nécessite des configurations spécifiques pour des structures complexes

09/12/2025

28

28

JAXB (Jakarta XML Binding), ou (Java Architecture for XML Binding)

- JAXB permet de convertir des données XML en objets Java et vice versa.
 - Il utilise des annotations pour mapper les éléments XML aux champs Java (sérialisation) et restaurer ces champs Java depuis XML (désérialisation)
- **Avantages :**
 - **Simplicité** : Élimine le besoin de parsing manuel en générant automatiquement des classes Java à partir de schémas XML.
 - **Productivité** : Facilite le travail avec des données XML en tant qu'objets Java.

09/12/2025

29

29

JAXB (Jakarta XML Binding)

Java Architecture for XML Binding (JAXB)

<https://www.oracle.com/technical-resources/articles/javase/jaxb.html>

- Il est idéal pour la sérialisation et la désérialisation XML, car il élimine la manipulation directe des nœuds ou des documents XML
- Les utilisateurs travaillent directement avec des **objets Java**, sans se soucier des détails de manipulation des fichiers XML.
- Recommandé si le projet contient des objets métier complexes.

09/12/2025

30

30

JAXB (Jakarta XML Binding)

- L'API JAXB propose un framework composé de classes regroupées dans trois packages :
- `javax.xml.bind` : contient les interfaces principales et la classe `JAXBContext`
- `javax.xml.bind.util` : contient des utilitaires
- `javax.xml.bind.helper`: contient une implémentation partielle de certaines interfaces pour faciliter le développement d'une implémentation des spécifications de JAXB

09/12/2025

31

31

Comment ça marche ? JAXB

- Utiliser des annotations sur les classes Java (`@XmlRootElement`, `@XmlElement`, etc.) pour gérer le processus de manière transparente.
- Définir une classe Java, et grâce aux annotations JAXB (comme `@XmlRootElement` et `@XmlElement`), vous indiquez comment l'objet doit être transformé en XML ou reconstruit depuis XML.
- La sérialisation et la désérialisation sont effectuées automatiquement via des méthodes de JAXB comme `marshal()` (pour écrire) et `unmarshal()` (pour lire).

09/12/2025

32

32

Du Concept au Code JAXB

ANALOGIE MEUBLATEX	CODE JAVA/JAXB
Meuble monté	(ex: <code>Student</code>)
Démontage en pièces	(method <code>marshal()</code>)
Carton avec pièces	(ex: <code>student.xml</code>)
Remontage du meuble	(method <code>unmarshal()</code>)
Notice IKEA	(<code>@XmlRootElement,</code> <code>@XmlElement</code>)

09/12/2025

33

33

Étape 1 : Créer une Classe Java

- Définir un **objet Java** (le meuble monté).

```
public class Student {
    private String name;
    private int age;
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
```

C'est juste une classe Java normale. JAXB va la "décorer" avec des annotations.

09/12/2025

34

34

Étape 2 : Ajouter les Annotations JAXB

- Indiquer à JAXB **comment faire la notice du meuble.**

```
@XmlRootElement public class Student {
    private String name;
    private int age;
    @XmlElement public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    @XmlElement public int getAge() { return age; } public
    void setAge(int age) { this.age = age; } }
```

@XmlRootElement = Cette classe est la racine du XML

@XmlElement = Ces propriétés deviennent des balises XML

09/12/2025

35

35

Étape 3 : Créer les Objets

- Instancier l'objet et **le remplir de données.**

```
// Créer un étudiant (meuble) Student student
= new Student(); student.setName("Alice");
student.setAge(22);
System.out.println("Étudiant créé en mémoire
✓");
System.out.println("Nom: " +
student.getName());
System.out.println("Âge: " +
student.getAge());
```

- À ce stade, l'étudiant n'existe que **en mémoire RAM**. Si on éteint l'ordinateur, tout est perdu !

09/12/2025

36

36

Étape 4 : Sérialiser avec marshal()

- Transformer l'objet Java en **fichier XML** (le démontage + carton).

```
// Créer un contexte
JAXB JAXBContext context =
JAXBContext.newInstance(Student.class);
// Créer un marshaller (l'outil de démontage)
Marshaller marshaller = context.createMarshaller();
marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
// Démontage : écrire le fichier XML
marshaller.marshal(student, new File("student.xml"));
System.out.println("✓ Fichier student.xml créé !");
```

- Le meuble est maintenant dans un carton sur le disque dur !

09/12/2025

37

37

Résultat : Le Fichier XML

- Voici ce qui a été écrit dans **student.xml** :

```
<?xml version="1.0" encoding="UTF-8"?>
<student>
<name>Ali</name> <age>22</age>
</student>
```

- C'est l'objet Student "à plat". On peut :

- L'envoyer par réseau
- Le sauvegarder sur disque
- L'ouvrir dans n'importe quel éditeur de texte
- Le partager avec Python /C#/ etc.

09/12/2025

38

38

Étape 5 : Désérialiser avec unmarshal()

- Transformer le fichier XML en **objet Java** (le remontage).

```
// Créer un contexte JAXB (même qu'avant)
JAXBContext context =
JAXBContext.newInstance(Student.class);
// Créer un unmarshaller (l'outil de remontage)
Unmarshaller unmarshaller =
context.createUnmarshaller();
// Remontage : lire le fichier XML et créer l'objet
Object obj = unmarshaller.unmarshal(new
File("student.xml")); Student studentLoaded =
(Student) obj; System.out.println("✓ Fichier rechargé
!"); System.out.println("Nom: " +
studentLoaded.getName()); System.out.println("Âge: " +
studentLoaded.getAge());
```

- Le meuble est remonté ! On peut l'utiliser normalement.

09/12/2025

39

39

Le Cycle Complet en Schéma

Java Object
En mémoire

marshal()
Sérialisation

XML File
Sur disque

⬇ Plus tard... ⬇

XML File

Chargé

unmarshal()

Déserialisation

Java Object

Restauré

09/12/2025

40

40

Les éléments du Cycle Complet

1. Les Annotations sont Obligatoires

Sans @XmlRootElement et @XmlElement, JAXB ne saura pas comment faire la conversion.

2. Getters/Setters Requis

JAXB utilise les getters/setters, pas d'accès direct aux champs privés.

3. Dépendances Java 11+

JAXB a été retiré du JDK. Vous devez ajouter jakarta.xml.bind-api dans votre pom.xml

4. Gestion des Erreurs

Utilisez try/catch pour capture JAXBException en cas d'erreur de conversion.

09/12/2025

41

41

Sérialisation XML

- La sérialisation XML sérialise uniquement les **champs publics et les valeurs de propriété** d'un objet dans un flux de données XML.
- La sérialisation **XML n'inclut pas d'informations de type**.
 - Par exemple, si un objet **Book** se trouve dans l'espace de noms **Library**, il n'y a aucune garantie qu'il soit déserialisé dans un objet du même type.
- La sérialisation XML ne convertit pas les méthodes, les indexeurs, les champs privés ni les propriétés en lecture seule

09/12/2025

42

42

<p>Partie 2</p> <pre> public class Main { public static void main(String[] args) throws Exception { // Création d'un objet Student Student student = new Student(); student.setName("ali"); student.setAge(22); // Sérialisation en XML JAXBContext context = JAXBContext.newInstance(Student.class); Marshaller marshaller = context.createMarshaller(); marshaller.setProperty(Marshaller.JAXB_FORMAT TED_OUTPUT, true); marshaller.marshal(student, new File("student.xml")); System.out.println("Sérialisation XML terminée !"); } } </pre>	<p>Partie 1</p> <pre> @XmlElement public class Student { private String name; } </pre> <p> • Crée une instance de JAXBContext spécifiquement pour la classe Student. • Ici, Student.class représente la classe Java que vous voulez lier à un document XML. Cette classe doit être correctement annotée avec les annotations JAXB (@XmlElement, @XmlElement, etc.) • Cette étape prépare JAXB à comprendre comment mapper les propriétés de la classe Student vers des éléments/attributs XML </p> <p> context.createMarshaller() : • Crée une instance de Marshaller à partir du JAXBContext. • Le Marshaller utilise les informations fournies par JAXBContext pour transformer l'objet Java en XML. </p> <p> setProperty() : • Configure une propriété du Marshaller. Ici, on indique que la sortie XML doit être "formatée". • JAXB_FORMATTED_OUTPUT : Une constante qui indique si le document XML généré doit être lisible par un humain (formaté avec des indentations et des sauts de ligne). • Valeur true : Active le formatage du document XML </p>
<p>09/12/2025</p>	<p>43</p>

<pre> marshaller.marshal(student, new File("student.xml")); </pre> <ul style="list-style-type: none"> Méthode utilisée pour effectuer la sérialisation d'un objet Java en XML. Paramètres : <ul style="list-style-type: none"> student : L'objet Java à sérialiser en XML. Cet objet doit être une instance de la classe Student (annotée avec JAXB). new File("student.xml") : Spécifie le fichier de destination où le document XML sera écrit. Ici, le fichier student.xml sera créé dans le répertoire de travail actuel (si le fichier n'existe pas) ou écrasé s'il existe déjà. JAXB_FORMATTED_OUTPUT : booléen qui indique si le document XML doit être formaté Processus : <ul style="list-style-type: none"> JAXB parcourt les propriétés de l'objet student. Chaque propriété est convertie en élément ou attribut XML selon les annotations JAXB de la classe Student. Le document XML est ensuite écrit dans le fichier student.xml. Rôle dans la sérialisation : <ul style="list-style-type: none"> C'est l'étape finale où l'objet Java est transformé en document XML et sauvegardé dans un fichier.

09/12/2025 44

Résumé du processus

- Initialisation du contexte (JAXBContext) : Prépare JAXB pour comprendre la classe xxxxxxx.
- Création du marshaller : Génère un outil capable de convertir l'objet Java en XML.
- Configuration du marshaller : Ajout d'options comme le formatage de sortie.
- Exécution de la sérialisation : Conversion de l'objet xxxxxxx en XML et sauvegarde dans un fichier

09/12/2025

45

45

Résumé des annotations clés et leur utilisation

Annotation

Utilisation principale

Portée

<code>@XmlRootElement</code>	Déclare une classe comme racine XML	Classe
<code>@XmlAttribute</code>	Définit l'ordre des champs dans XML	Classe
<code>@XmlElement</code>	Mappe un champ/propriété à un élément XML	Champ/Getter
<code>@XmlAttribute</code>	Mappe un champ/propriété à un attribut XML	Champ/Getter
<code>@XmlTransient</code>	Exclut un champ/propriété de la sérialisation	Champ/Getter
<code>@XmlAccessorType</code>	Contrôle la stratégie d'accès	Classe
<code>@XmlElementWrapper</code>	Regroupe une collection dans un conteneur XML	Champ/Getter
<code>@XmlJavaTypeAdapter</code>	Personnalise le mapping entre Java et XML	Champ/Getter

09/12/2025

46

46

Avantages JAXB

- **Interopérabilité**

- JAXB est intégré dans la plateforme Java SE, ce qui le rend facilement accessible sans nécessiter de bibliothèques externes.
- Il supporte une large gamme de types d'entrée/sortie (fichiers, flux, nœuds DOM, etc.)

- **Génération de classes à partir de schémas :**

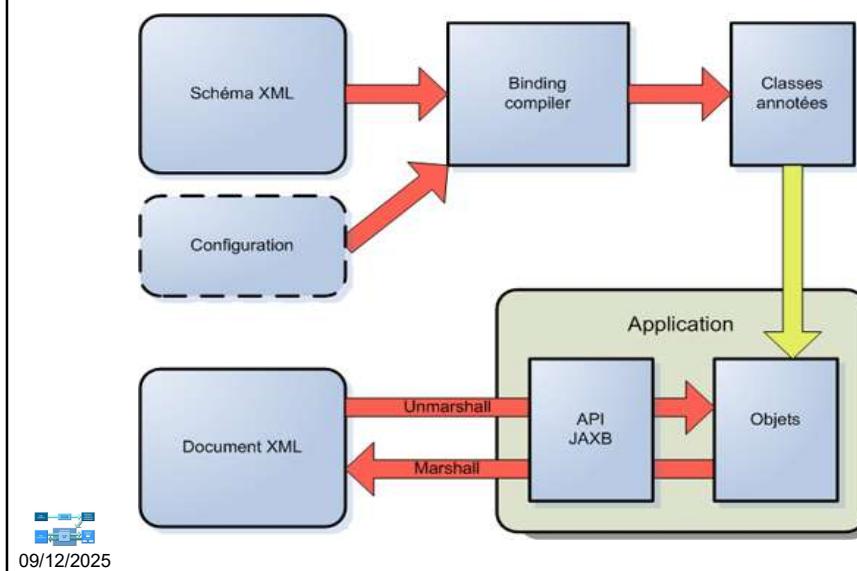
- JAXB peut générer des classes Java à partir de schémas XML (XSD), ce qui assure une correspondance précise entre le schéma et les objets Java
- Cela permet de garantir que les documents XML respectent les contraintes définies par le schéma.

09/12/2025

47

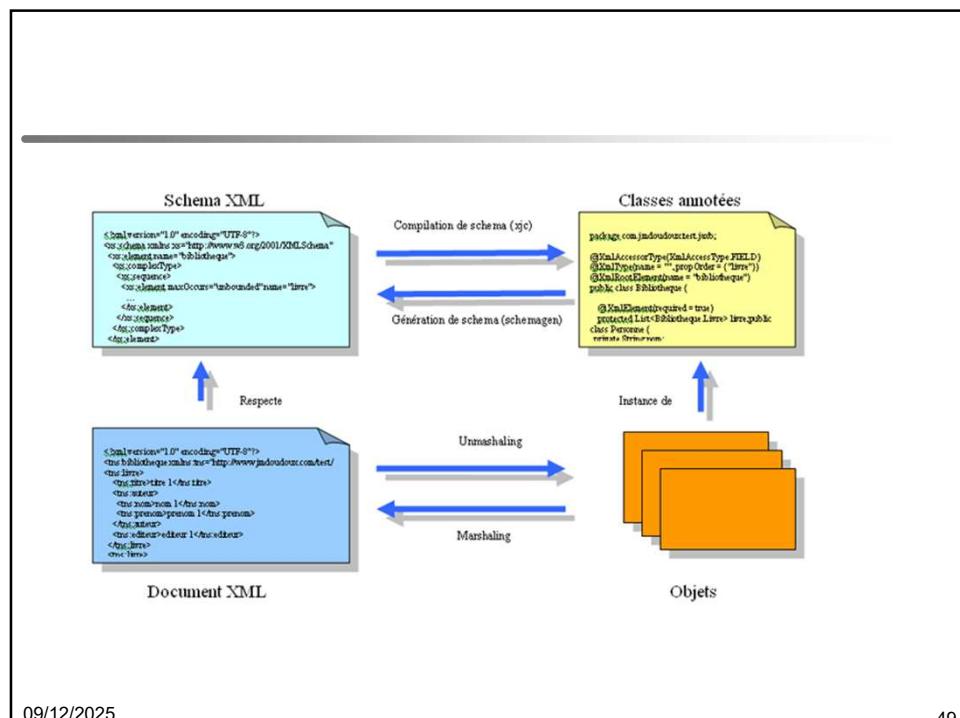
47

Génération de classes à partir de schémas



48

48



09/12/2025

49

49

Génération de classes à partir de schémas

- JAXB permet de générer des classes Java à partir de schémas XML (XSD). Cela garantit que les documents XML respectent les contraintes définies par le schéma.

Étapes pour Générer des Classes JAXB à partir d'un Schéma XML (XSD) : 1. Ecrire un schéma .xsd

```

2  <!--employee.xsd-->
3  <xsschema xmlns:xss="http://www.w3.org/2001/XMLSchema">
4    <xselement name="employee">
5      <xsccomplexType>
6        <xsssequence>
7          <xselement name="id" type="xsi:int"/>
8          <xselement name="name" type="xsi:string"/>
9          <xselement name="department" type="xsi:string"/>
10         </xsssequence>
11       </xsccomplexType>
12     </xselement>
13   </xsschema>

```

09/12/2025

50

50

Génération de classes à partir de schémas

2. Utiliser l'Outil XJC pour Générer les Classes Java :

L'outil XJC (XML to Java Compiler) transforme le schéma XSD en classes Java annotées pour JAXB.

Exécution de la commande Basic: `xjc employee.xsd`

`xjc -d src -p com.example.jaxb employee.xsd`

-d src : Spécifie le répertoire de destination pour les classes générées.
-p com.example.jaxb : Spécifie le package pour les classes générées.

Option	Rôle
-p nom_package	Précise le package qui va contenir les classes générées
-d répertoire	Précise le répertoire qui va contenir les classes générées
-nv	Inhibe la validation du schéma
-b fichier	Précise un fichier de configuration
-classpath chemin	Précise le classpath

09/12/2025

51

51

Génération de classes à partir de schémas

3. Classes Java Générées :

Les classes générées contiennent des annotations JAXB pour la sérialisation et la désérialisation XML.

4. Sérialisation et Désérialisation avec JAXB :

Utiliser les classes générées pour sérialiser et désérialiser des objets Java en XML et vice versa.

```
// Employee.java
package com.example.jaxb;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement
public class Employee {
    private int id;
    private String name;
    private String department;
    @XmlElement
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    @XmlElement
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @XmlElement
    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
        this.department = department;
    }
}
```

09/12/2025

52

Applications simples

- Sérialisation simple d'un objet "Book" ...
- Exercice : Sérialisation d'un "Library" contenant plusieurs "Book"
- Exercice : Génération de classes Java depuis un XSD
- Exercice : Modifier le nom des éléments XML avec un fichier de liaison

09/12/2025

53

53

Sauvegarde d'État de Jeu avec JAXB

- **Exemple** : Imaginez un jeu vidéo où le joueur peut sauvegarder sa progression. La sérialisation permet de convertir l'état actuel du jeu (position du joueur, score, inventaire, etc.) en un format XML. Ce fichier XML peut ensuite être stocké sur le disque et rechargé plus tard pour restaurer l'état du jeu.
 - **Sérialisation** : Convertir les objets du jeu (joueur, ennemis, objets) en XML.
 - **Stockage** : Enregistrer le fichier XML sur le disque.
 - **Désérialisation** : Lire le fichier XML et recréer les objets du jeu pour restaurer l'état.
- **Définir la Classe GameState :**
 - Annoter la classe et ses champs avec les annotations JAXB pour indiquer comment ils doivent être sérialisés.
- **Sérialiser l'État du Jeu :**
 - Utiliser JAXB pour convertir l'objet GameState en XML et l'enregistrer dans un fichier.
- **Désérialiser l'État du Jeu :**
 - Utiliser JAXB pour lire le fichier XML et recréer l'objet GameState.

09/12/2025

54

54

JAXB résumé

- support des schémas XML et mise en œuvre des annotations
- assure la correspondance bidirectionnelle entre un schéma XML et le bean correspondant.
- Être configurable : JAXB met en œuvre des fonctionnalités par défaut qu'il est possible de modifier par configuration pour répondre à ses propres besoins
- S'assurer que la création d'un document XML à partir d'objets et retransformer ce document en objets donne le même ensemble d'objets
- Pouvoir valider un document XML ou les objets qui encapsulent un document sans avoir à écrire le document correspondant

09/12/2025

55

55

JAXB résumé

- L'utilisation de JAXB implique généralement deux étapes :
 - Génération des classes et interfaces à partir du schéma XML
 - Utilisation des classes générées et de l'API JAXB pour transformer un document XML en graphe d'objets et vice versa, pour manipuler les données dans le graphe d'objets et pour valider le document

09/12/2025

56

56

- JAXB 2.0 est incorporée dans Java EE 5 et dans Java SE 6.
- Avec la version fournie avec JWSDP 2.0, les bibliothèques suivantes doivent être ajoutées au classpath :
jaxb\lib\jaxb-api.jar,
jaxb\lib\jaxb-impl.jar,
jaxb\lib\jaxb-xjc.jar,
jwsdp-shared\lib\activation.jar,
sjsxp\lib\jsr173_api.jar,
sjsxp\lib\sjsxp.jar
- JAXB 2.0 requiert un JDK 5.0 minimum pour être utilisé.

09/12/2025

57

57

Bonnes pratiques

1. Gestion des Erreurs

- **Validation des Données** : Toujours valider les données avant la sérialisation pour éviter les erreurs de format et les données incorrectes.
- **Gestion des Exceptions** : Utiliser des blocs try-catch pour gérer les exceptions lors de la sérialisation et de la désérialisation. Fournir des messages d'erreur clairs et informatifs.
- **Journalisation** : Enregistrer les erreurs et les exceptions dans des fichiers de log pour faciliter le débogage et la maintenance.

09/12/2025

58

58

```
try {
    JAXBContext context =
JAXBContext.newInstance(GameState.class);
    Marshaller marshaller = context.createMarshaller();
    marshaller.marshal(gameState, new File("gameState.xml"));
} catch (JAXBException e) {
    e.printStackTrace();
    // Log the error
}
```

09/12/2025

59

59

• 2. Sécurité

- **Validation des Schémas** : Valider les documents XML contre un schéma XSD pour s'assurer qu'ils respectent les contraintes définies.
- **Éviter les Entités Externes** : Désactiver la résolution des entités externes pour prévenir les attaques de type XXE (XML External Entity).

```
Unmarshaller unmarshaller = context.createUnmarshaller();
SchemaFactory sf =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
Schema schema = sf.newSchema(new File("gameState.xsd"));
unmarshaller.setSchema(schema);
unmarshaller.setProperty(XMLConstants.ACCESS_EXTERNAL_DTD, "");
unmarshaller.setProperty(XMLConstants.ACCESS_EXTERNAL_SCHEMA,
"");
```

09/12/2025

60

60

- **3. Performance**

- **Optimisation des Objets** : Éviter de sérialiser des objets volumineux ou inutiles. Utiliser des structures de données appropriées pour minimiser la taille des documents XML.
- **Compression** : Envisager de compresser les fichiers XML pour réduire l'espace de stockage et améliorer les temps de transmission.

- **4. Lisibilité et Maintenance**

- **Formatage** : Utiliser des options de formatage pour rendre les documents XML lisibles par les humains. Cela facilite le débogage et la maintenance.
- **Documentation** : Documenter les classes et les méthodes utilisées pour la sérialisation. Inclure des commentaires sur les annotations JAXB et leur utilisation.

```
marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,  
Boolean.TRUE);
```

09/12/2025

61

61

Conseils

1. Validation des Données : Utiliser les schémas XSD pour valider les documents XML avant la désérialisation.

2. Gestion des Erreurs : Implémenter une gestion robuste des exceptions pour gérer les erreurs de sérialisation et de désérialisation.

3. Documentation : Documenter les classes et les méthodes pour faciliter la maintenance et la compréhension du code.

4. Tests : Écrire des tests unitaires pour vérifier la sérialisation et la désérialisation des objets.

09/12/2025

62

62

- Lorsque votre schéma XSD contient plusieurs éléments globaux pouvant servir de racines potentielles, vous pouvez choisir l'élément racine pour chaque opération de sérialisation en spécifiant explicitement l'élément racine dans votre code
- En utilisant les annotations `@XmlRootElement` et en spécifiant le contexte JAXB approprié pour chaque opération de sérialisation et de désérialisation, vous pouvez choisir l'élément racine à utiliser. Cela permet de gérer efficacement les schémas XSD contenant plusieurs éléments globaux.

09/12/2025

63

63

Utilité des standards

- Capitaliser les efforts
 - Éviter une réécriture d'un interpréteur XML à chaque nouvelle application
- Produire du code indépendant de la plate-forme et du langage de programmation
 - Interopérabilité
- Deux principales API standardisées pour la manipulation de fichiers XML
 - SAX & **DOM** (intégrées dans Java1.4: JSAX, JDOM)
- Un langage d'expression de chemin dans le même document et entre les documents
 - **XPath**
- Autres langages assurant aussi la manipulation d'un document XML
 - XLink, XPointer, ...

09/12/2025

64

64

Approches d'API de lecture/écriture

- Manipulation d'arbres/graphes d'objets
 - Approche aisée : accès direct et aléatoire aux structures en mémoire
 - Approche coûteuse : les structures sont intégralement en mémoire
- Manipulation d'événements :
 - On ne considère pas la structure mais des événements élémentaires : ouverture d'un objet, d'un tableau, écriture d'une valeur, fermeture d'un objet...
 - Les structures ne résident pas en mémoire, les accès sont séquentiels

09/12/2025

65

65

Le « DOM »...

- Document Object Model : standard élaboré par le W3C
- Au départ : permettre aux scripts JavaScript et aux programmes Java de fonctionner sur différents browsers
- Standard à plusieurs niveaux, qui se présentent comme des couches fonctionnelles
 - Niveau 0 : Fonctions communes aux navigateurs «3.0», pas de spécification W3C.
 - Niveau 1 : Définit les fonctionnalités pour manipuler les informations de style
 - Niveau 2 : chargement et enregistrement de documents et de modèles de contenu comme «DTD»

09/12/2025

66

66

Dans le tp

Le « DOM »...

- API définie par la W3C avec de nombreuses implantations : actuellement version 3 (v4 en développement)
- Manipule le document XML sous la forme d'un arbre
- Permet d'associer aussi des événements aux noeuds
- Incontournable pour la manipulation de pages HTML avec JavaScript (mais ce n'est pas l'objet de ce cours)
- API présente dans le JDK Java (org.w3c.dom)

09/12/2025

67

67

Dans le tp

Principes du DOM

- Créer un graphe d'objets = instance en mémoire du document
- Faciliter l'accès et la modification de contenus et structures des documents XML
- DOM définit de manière standardisée :
 - L'ensemble d'objets pour représenter l'instance du document en mémoire
 - L'organisation de ces objets sous forme arborescente
 - Les interfaces de ces objets : méthodes et attributs exposés

09/12/2025

68

68

Dans le tp

Interfaces du DOM

- Les objets de programmation représentant le document XML sont des nœuds
 - Un nœud pour chaque composant de base du document XML (élément, attribut, instruction...)
 - Un type de nœud pour chaque type de composants
- La racine du document est représentée par l'interface « `node` »: Nœud de l'arbre DOM, ancêtre commun à tous les types de nœuds
 - le contenu est représenté par les interfaces « `document` », « `element` », « `attribute` », « `characterData` », et peut contenir aussi
 - « `instruction de traitement`», « `commentaire` », « `Cdata` », « `characterData` »

09/12/2025

69

69

Dans le tp

- **Les principaux noeuds**
- Ils implantent l'interface `org.w3c.dom.Node`

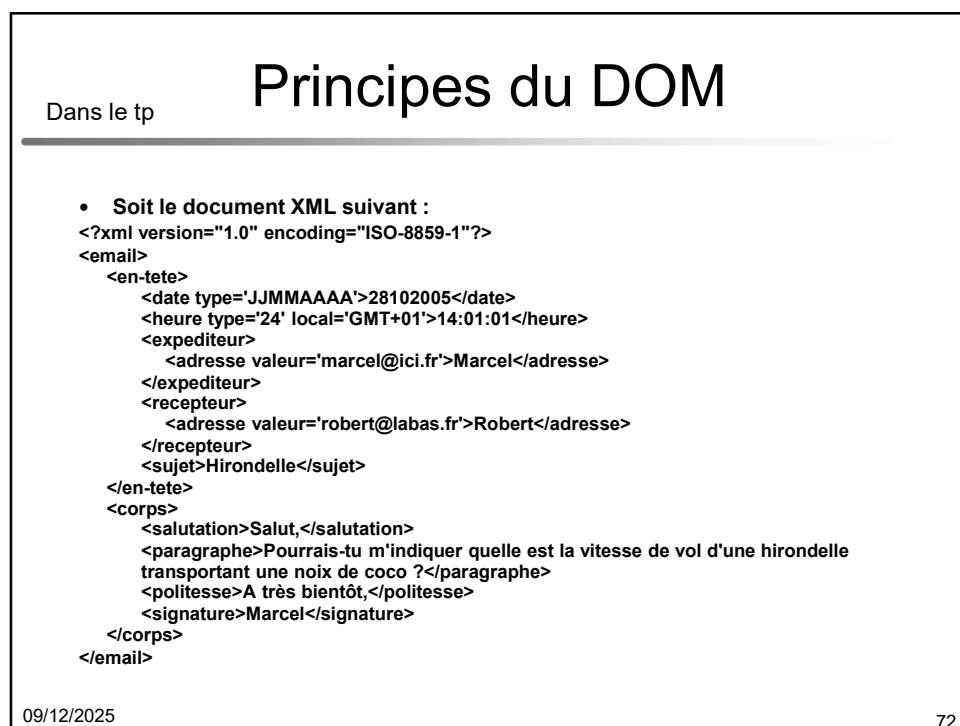
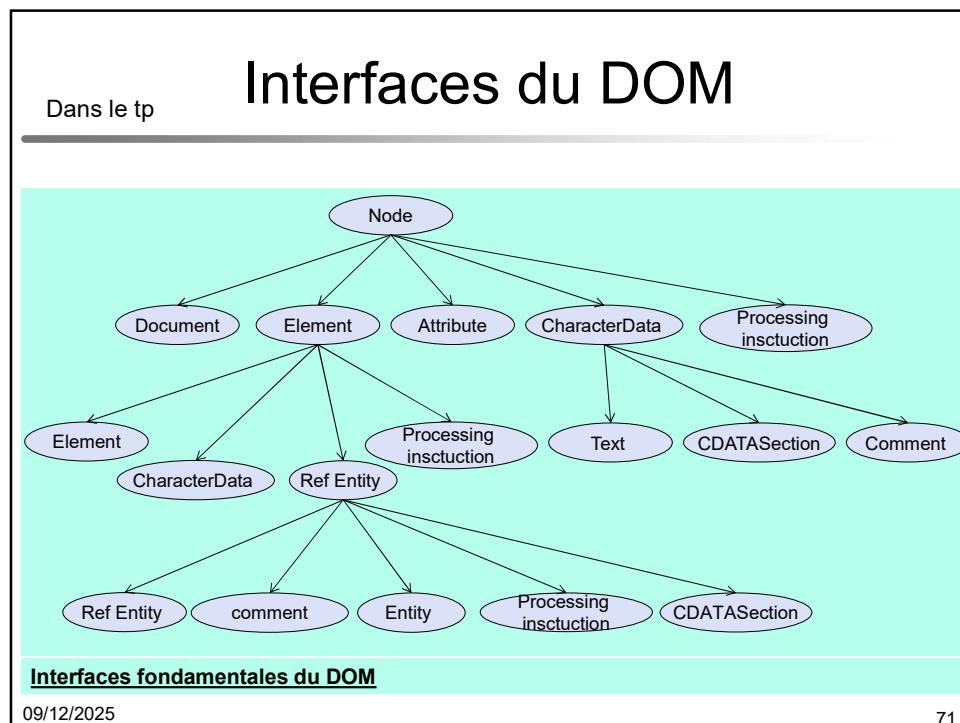
Noeuds	Enfants possibles
<code>Document</code>	<code>Element, ProcessingInstruction, Comment, DocumentType</code>
<code>DocumentFragment</code>	
<code>Element</code>	<code>Element, Text, Comment, ProcessingInstruction, CDataSection, EntityReference</code>
<code>Attr</code>	<code>Text, EntityReference</code>
<code>Text</code>	Feuille
<code>CDataSection</code>	Feuille
<code>Comment</code>	Feuille

<code>ProcessingInstruction</code>	Feuille
<code>Entity</code>	<code>Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference</code>
<code>EntityReference</code>	
<code>Notation</code>	Feuille

09/12/2025

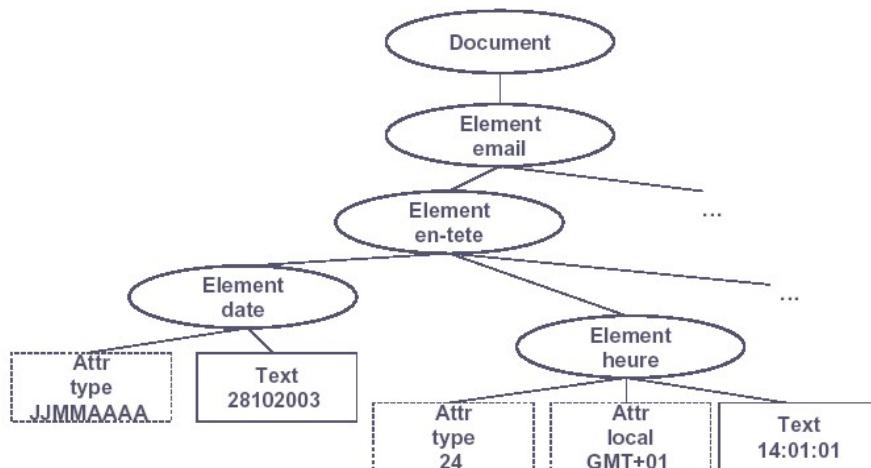
70

70



Dans le tp

Principes du DOM



09/12/2025

73

73

Propriétés des nœuds dans DOM...

Dans le tp

- **parentNode** the parent node of the current node
- **childNodes[n]** retrieves the nth child of the current node
- **firstChild** the first child of the current node, or nothing if there's none
- **lastChild** the last child of the current node, or nothing if there's none
- **nextSibling** the next sibling of the current node, or nothing if there's none
- **prevSibling** the previous sibling of the current node, or nothing if there's none
- **nodeType** the node type, such as "TR", "IMG", "TABLE", etc.
- **attributes** the node's attribute objects
- **nodeName** the name of the node
- **+nodeValue** (si le noeud est de type feuille : du texte)

09/12/2025

74

74

Dans le tp

- Informations sur un noeud
- String getNodeName() : nom de l'élément, de l'attribut
- String getNodeValue() : valeur textuelle contenue pour un attribut, du texte, un commentaire (null pour un élément)
- String getTextContent() : contenu textuel du noeud (en explorant le sous-arbre)
- short getNodeType() : constante pour le type de noeud (évite d'utiliser instanceof)
- NodeList getChildNodes() : tous les noeuds enfants
- NodeList getElementsByTagName(String tagName) : noeuds enfants portant un certain nom
- Node getParentNode() : noeud parent
- String getAttribute(String name) : attribut d'un élément
- Comment utiliser NodeList et NamedNodeMap ?
- En utilisant (entre-autres) les méthodes int getLength() et Node item(int i)

09/12/2025

75

75

Autres plateformes

Pour plus de connaissances

Hors cours

09/12/2025

76

76

Bibliothèque standard de Java (package java.beans)

```
// Exemple de code Java pour sérialiser l'état du jeu
public class GameState {    Déclare une classe publique appelée GameState, qui représente l'état du jeu
    private String playerName;    Déclaration de trois attributs privés pour la classe :
    private int score;    •playerName : le nom du joueur, une chaîne de caractères.
    private List<String> inventory;    •score : le score du joueur, un entier.
                                         •inventory : une liste de chaînes représentant l'inventaire du joueur
                                         (objets collectés dans le jeu).

    // faut ajouter des méthodes getter et setter pour accéder et modifier ces attributs. (Non
    montrés dans cet extrait mais nécessaires pour que XMLEncoder fonctionne.)
}

// Sérialisation de l'état du jeu
GameState gameState = new GameState("Alice", 1000, Arrays.asList("Sword", "Shield"));
try (FileOutputStream fos = new FileOutputStream("gameState.xml");
     XMLEncoder encoder = new XMLEncoder(fos)) {
    encoder.writeObject(gameState);
} catch (IOException e) {
    e.printStackTrace();
}
```

Une instance de GameState avec des données fictives :
•playerName est "Alice".
•score est 1000.
•inventory contient deux éléments : "Sword" et "Shield".
Note : Ce code suppose que la classe GameState possède un constructeur approprié prenant ces trois paramètres.

09/12/2025

77

77

Bibliothèque standard de Java (package java.beans)

```
// Sérialisation de l'état du jeu
GameState gameState = new GameState("Alice", 1000, Arrays.asList("Sword", "Shield"));
try (FileOutputStream fos = new FileOutputStream("gameState.xml");
     XMLEncoder encoder = new XMLEncoder(fos)) {
    encoder.writeObject(gameState);
} catch (IOException e) {
    e.printStackTrace();
}
```

Sérialise l'objet gameState en écrivant son état dans le fichier XML.

Capture les éventuelles exceptions d'entrée/sortie (par exemple, si le fichier ne peut pas être créé ou écrit) et imprime une trace de la pile d'erreurs.

09/12/2025

78

78

Bibliothèque standard de Java (package java.beans)

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <java>
3 <object class="GameState">
4 <void property="inventory">
5 <array class="java.lang.String" length="2">
6 <void index="0">
7 <string>Sword</string>
8 </void>
9 <void index="1">
10 <string>Shield</string>
11 </void>
12 </array>
13 </void>
14 <void property="playerName">
15 <string>Alice</string>
16 </void>
17 <void property="score">
18 <int>1000</int>
19 </void>
20 </object>
21 </java>
```

- Ce code crée un fichier nommé gameState.xml dans le répertoire de travail, contenant les données sérialisées de l'objet GameState.
- Voici un exemple de contenu du fichier XML généré

09/12/2025

79

79

Détails de l'outil XMLEncoder

- Fait partie du package java.beans depuis Java 1.4.
- Utilisé pour sérialiser des objets Java en XML de manière simple et rapide.
- Il génère un fichier XML qui peut être lu par son homologue XMLDecoder pour déserialiser l'objet.
- Contrairement à des outils plus avancés comme JAXB ou Jackson, XMLEncoder est conçu pour des scénarios de sérialisation simples et ne nécessite pas de configuration supplémentaire.

09/12/2025

80

80

Détails de l'outil XMLEncoder

- Caractéristiques de XMLEncoder
 - Pas besoin de bibliothèques tierces : il est directement inclus dans le JDK.
 - Fonctionne bien pour des objets qui respectent les conventions JavaBeans (constructeur sans paramètres, getters et setters publics).
- Limité dans les scénarios avancés, comme la sérialisation de structures complexes ou l'intégration avec des schémas XML (XSD).

09/12/2025

81

81

Sérialisation XML en C# .NET

- La classe centrale pour la sérialisation XML en « .net » est XmlSerializer. Les méthodes les plus importantes de cette classe sont Serialize et Deserialize

```
// Exemple de code C# .net pour sérialiser l'état du jeu
Classe GameState ,,
[Serializable] // Attribut Obligatoire pour indiquer que la classe peut être sérialisée.

// Méthode pour sérialiser un objet en XML

static void SerializeToXML(GameState gameState, string filePath)
{
    XmlSerializer serializer = new XmlSerializer(typeof(GameState));
    using (StreamWriter writer = new StreamWriter(filePath))
    {
        serializer.Serialize(writer, gameState);
    }
}
```

La méthode SerializeToXML utilise la classe XmlSerializer pour convertir un objet en XML et écrire ce XML dans un fichier.

09/12/2025

82

82

- Résultat attendu : Lorsque ce code est exécuté, il génère un fichier GameState.xml contenant le XML suivant :

```
1 <?xml version='1.0' encoding='utf-8'?>
2 <GameState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3   <PlayerName>Alice</PlayerName>
4   <Score>1000</Score>
5   <Inventory>
6     <string>Sword</string>
7     <string>Shield</string>
8   </Inventory>
9 </GameState>
```

La méthode `DeserializeFromXML` utilise la classe `XmlSerializer` pour lire un fichier XML et recréer un objet GameState.

// Méthode pour déserialiser un objet depuis un fichier XML

```
static GameState DeserializeFromXML(string filePath)
{
    XmlSerializer serializer = new XmlSerializer(typeof(GameState));
    using (StreamReader reader = new StreamReader(filePath))
    {
        return (GameState)serializer.Deserialize(reader);
    }
}
```

09/12/2025

83

83

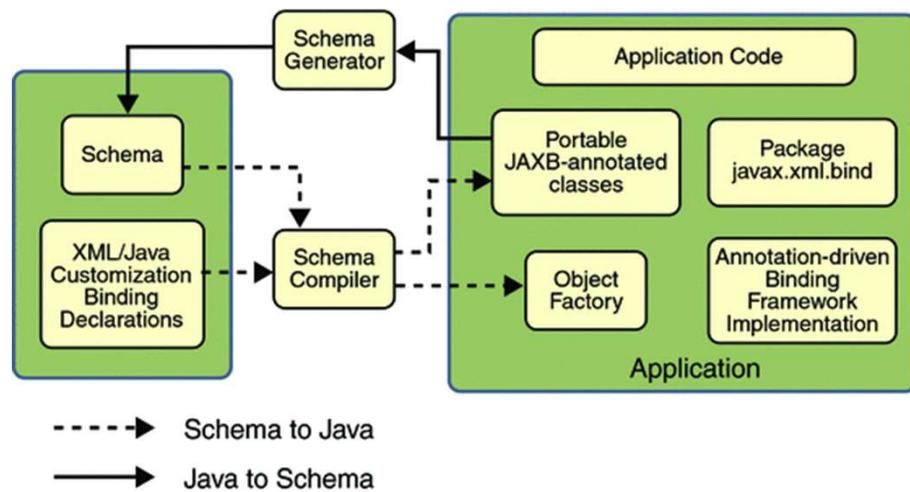
- En dehors de C#
- Pour des besoins plus complexes en Java :
 - le respect d'un schéma XML
 - les transformations avancées,
- Des outils comme JAXB (Java Architecture for XML Binding) ou Jackson XML sont plus adaptés.
 - Ils permettent une personnalisation avancée
 - Ils prennent mieux en charge des structures plus sophistiquées.

09/12/2025

84

84

Génération de classes à partir de schémas



09/12/2025

85

85