HOPEX UML - XML Schemas

HOPEX V2

HOPEX UML

Guide d’utilisation

HOPEX V2

A PROPOS D'HOPEX UML

Le langage de modélisation UML (Unified Modeling Language) s'est érigé comme standard pour la modélisation graphique des systèmes d'information. **HOPEX UML** offre un ensemble de vues et d'outils vous permettant de modéliser votre SI conformément à la version 2.3 de ce standard.

Ce guide a pour objectif de vous faire découvrir les principales fonctionnalités d'HOPEX UML.

CONVENTIONS UTILISÉES DANS LE GUIDE

☛ *Remarque sur les points qui précèdent.*

📖 *Définition des termes employés.*

☺ *Astuce qui peut faciliter la vie de l'utilisateur.*

Compatibilité avec les versions précédentes.

Ce qu'il faut éviter de faire.

Remarque très importante à prendre en compte pour ne pas commettre d'erreurs durant une manipulation.

Les commandes sont présentées ainsi : **Fichier > Ouvrir**.

Les noms de produits et de modules techniques sont présentés ainsi : **HOPEX**.

PRÉSENTATION

HOPEX UML vous permet de :

Analyser les cas d'utilisation

Une réflexion sur les fonctionnalités attendues du futur système est nécessaire avant sa conception. Les composants de ce système vont en effet être utilisés par les acteurs de l'organisation pour effectuer leur mission. Les divers "cas d'utilisation" de ce système vont être présentés dans les ***diagrammes de cas d'utilisation***.

Ils serviront de point de départ pour la découverte des objets.

Ils permettront ensuite de valider l'utilisation de ces objets à l'aide des diagrammes d'interaction.

Ils donneront enfin des critères de regroupement en "paquetage" pour les objets découverts.

Identifier les objets

Les objets ayant une structure semblable, le même comportement et les mêmes types de relations avec d'autres objets sont réunis dans une classe.

Le ***diagramme de classes*** permet d'identifier les objets mis en jeu à l'intérieur du système et de définir leur structure en termes d'attributs et d'opérations ainsi que les relations entre eux. Le ***diagramme d'objets*** montre les instances compatibles avec un diagramme de classes particulier qu'on peut ainsi valider avec un exemple.

Décrire les comportements

Le ***diagramme de machine à états*** permet de définir le comportement d'un objet vis-à-vis des sollicitations internes ou externes auxquelles il peut être soumis. Chacun des états dans lequel peut se trouver l'objet est indiqué ainsi que les réactions de l'objet à un événement donné en fonction de l'état où il se trouve.

Le ***diagrammes d'activité*** décrit également un comportement, mais en termes d'actions.

Représenter les interactions entre les objets

Le dialogue qui s'instaure entre les différents objets concernés par l'événement pour y répondre peut être représenté dans des ***diagrammes d'interaction***.

Les diagrammes d'interaction insistent sur les échanges qui ont lieu entre les objets.

Le ***diagramme de séquence*** présente ces mêmes échanges en mettant en évidence leur chronologie.

Le ***diagramme de communication*** met l'accent sur l'organisation structurelle des objets qui envoient et reçoivent des messages.

Le ***diagramme de vue générale d'interaction*** donne une vue d'ensemble du flot de contrôle.

Répartir les classes entre les paquetages

Une fois les objets identifiés, il est aisé de répartir les classes qui les implémentent entre les différents paquetages. Les regroupements de ces classes effectués dans le ***diagramme de paquetages*** sont faits de manière à minimiser les échanges entre les différents paquetages. Ils obéissent à deux critères : l'un, technique, lié à leur environnement d'exécution et l'autre, organisationnel, lié à l'emploi qui en sera fait par les différents utilisateurs à équiper pour chaque cas d'utilisation.

Définir les interfaces

Pour respecter le principe d'encapsulation, la répartition des éléments entre les composants est stricte. Il est donc nécessaire de prévoir les interfaces entre les éléments ayant des relations entre eux et qui appartiennent cependant à des composants différents.

Le ***diagramme de composants*** et le ***diagramme de structure composite*** présentent l'interdépendance des composants ou éléments d'un composant.

La définition des interfaces des objets ainsi que l'adhésion à un protocole d'échange normalisé (CORBA2, DCOM/OLE) sont des facteurs clés de l'interopérabilité, c'est-à-dire la capacité à faire coopérer les objets développés et exploités dans des environnements hétérogènes.

Spécifier la mise en oeuvre

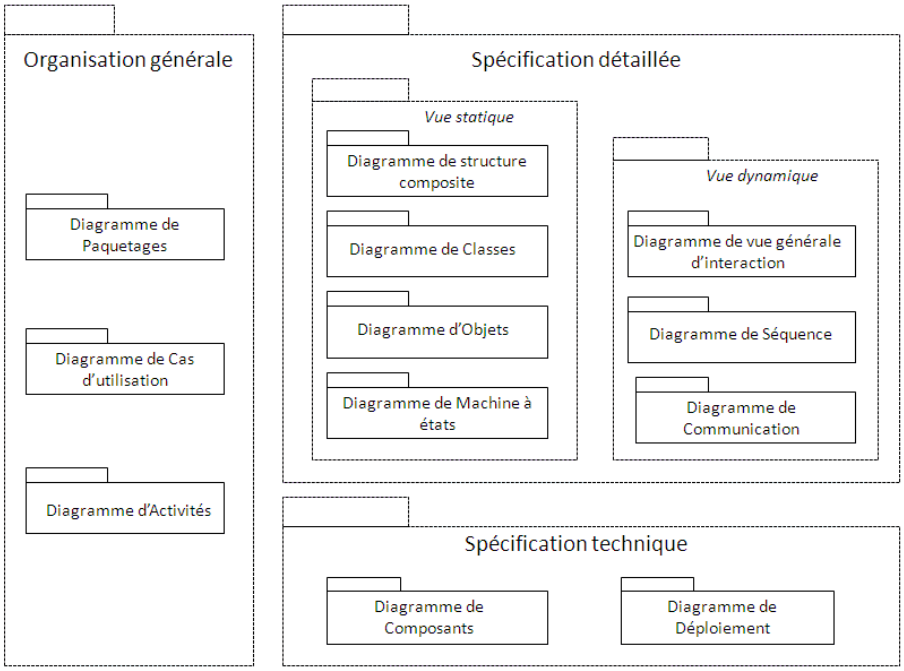
La mise en oeuvre des objets dans un environnement de travail concret peut être spécifiée les ***diagrammes de déploiement***.

ORGANISATION DES DIAGRAMMES D'UML

Organisation générale

Les ***diagrammes de cas d'utilisation*** vous permettent de montrer les principales interactions entre le système étudié et son environnement, et de mettre en évidence ses principaux sous-systèmes.

Les ***diagrammes de paquetages*** constituent un découpage du système. Le découpage du système en paquetages est relativement structurant dans la mesure où un objet ne peut être que dans un seul paquetage. Vous pouvez commencer à dessiner des diagrammes de paquetages dès que vous avez identifié les principaux composants de votre système (Commercial, Production, Facturation, …).



Spécification détaillée

Le diagramme principal est le ***diagramme de classes***. Il décrit l'essentiel de la sémantique des objets composant le système. C'est le diagramme sur lequel les concepteurs passeront généralement le plus de temps. La découverte des classes se fait généralement par des aller-retour entre les diagrammes de classes et les diagrammes de séquence.

Le ***diagramme de machine à états*** permet de décrire l'aspect statique d'un objet, c'est à dire les différents états dans lesquels il peut se trouver et les transitions possibles entre ces états. Il permet ainsi de compléter la description d'une classe.

Les ***diagrammes d'interaction*** permettent de spécifier l'aspect dynamique du système en montrant l'interaction des objets entre eux. Ils permettent en particulier de décrire de façon détaillée les différents scénarios de fonctionnement d'un cas d'utilisation. Le diagramme de séquence explicite plutôt le déroulement d'un scénario dans le temps, tandis que le diagramme de communication insiste plutôt sur l'interaction entre les objets.

Spécification technique et mise en oeuvre

Le ***diagramme de composants*** décrit les différents composants techniques d'une application et leurs interactions.

Le ***diagramme de structure composite*** précise les collaborations entre les composants ou éléments d'un composant dans l'exécution d'une tâche commune.

Le ***diagramme de déploiement*** permet de préciser l'architecture technique du système en indiquant sur quels postes de travail ou sur quels nœuds du système informatique seront installés les différents composants de l'application.

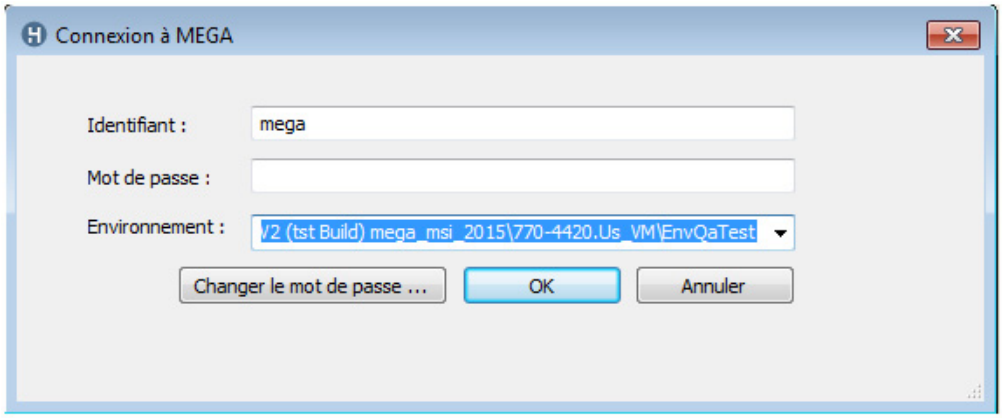
Points d'entrée des diagrammes d'UML

Diagramme	**Points d'entrée**
Diagramme de classes	Paquetage, Classe, Cas d'utilisation
Diagramme d'objets	Classe, Composant, Paquetage, Cas d'utilisation
Diagramme de composants	Composant, Paquetage
Diagramme de structure composite	Composant, Classe, Collaboration
Diagramme de déploiement	Paquetage
Diagramme de paquetages	Paquetage, Bibliothèque
Diagramme de cas d'utilisation	Paquetage, Cas d'utilisation
Diagramme de séquence (UML2)	Interaction
Diagramme de communication	Interaction
Diagramme de vue générale d'interaction	Interaction
Diagramme d'activités (UML2)	Activité
Diagramme de machine à états	Machine à état, Machine à état de protocole

LANCER HOPEX UML

Pour lancer **HOPEX** :

1. Double-cliquez sur l'icône **HOPEX** mega.exe .
   La fenêtre d'identification apparaît.



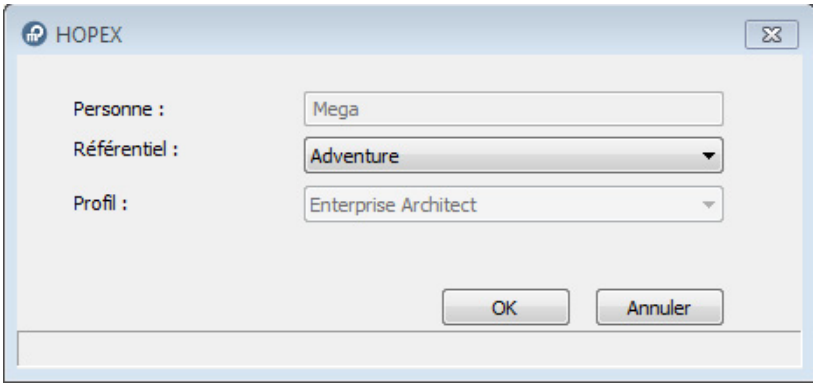
2. Dans le champ **Environnement**, sélectionnez votre environnement de travail.

   ☛ *Si vous n'avez accès qu'à un environnement, celui–ci est automatiquement pris en compte et le champ de sélection de l'environnement est grisé.*

   ☛ *Pour en savoir plus sur les environnements et les utilisateurs, voir les chapitres **Gérer les environnements** et **Gérer les utilisateurs** du **HOPEX Administration - Supervisor**.*

3. Dans le champ **Identifiant**, saisissez votre identifiant.
4. Dans le champ **Mot de passe**, saisissez votre mot de passe (si nécessaire).

5. Cliquez sur **OK**.
Lorsque vous êtes authentifié, la fenêtre de définition de votre espace de travail apparaît.



Le champ **Personne** est défini automatiquement ; il indique le nom de la personne associée à l'identifiant renseigné dans la fenêtre de connexion.

6. Dans le champ **Référentiel**, sélectionnez votre référentiel de travail.

> ☛ *Le menu déroulant vous permet de faire apparaître la liste des référentiels disponibles dans l'environnement. Lorsque vous commencez à modéliser les données de votre entreprise, il est recommandé de créer un nouveau référentiel dans un nouvel environnement.*
>
> ☛ *Si vous n'avez accès qu'à un seul référentiel, celui–ci est automatiquement pris en compte.*

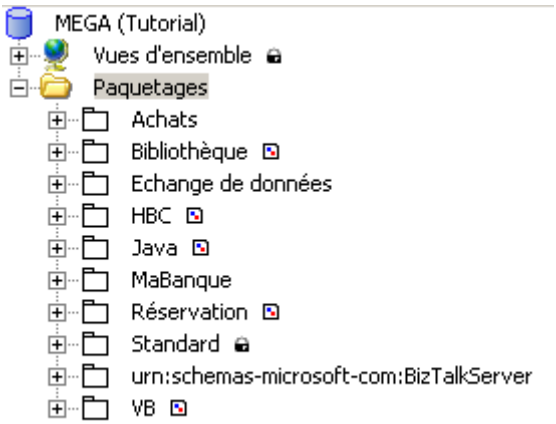
7. Dans le champ **Profil**, cliquez sur la flèche et sélectionnez le profil avec lequel vous voulez travailler.

> ☛ *Si vous n'avez qu'un profil, celui-ci est automatiquement pris en compte.*

8. Cliquez sur **OK**.
Votre bureau apparaît.

Découvrir l'espace de travail HOPEX UML

Dans le navigateur, cliquez sur le + devant "Paquetage" pour voir apparaître la liste des paquetages de premier niveau.



L'icône visible à côté de certains paquetages signifie qu'ils sont décrits par un diagramme.

LE DIAGRAMME DE CAS D'UTILISATION

Le diagramme de cas d'utilisation constitue une première étape dans la description d'un système d'information. Il permet d'identifier les fonctionnalités que doit fournir le système afin de répondre aux besoins des acteurs de l'organisation ; il décrit en ce sens les interactions entre le système et les acteurs.

Les points suivants sont abordés ici :

CRÉER UN DIAGRAMME DE CAS D'UTILISATION

Un diagramme de cas d'utilisation permet de décrire les interactions entre les acteurs de l'organisation et le système dans chacun des *cas d'utilisation* envisagés.

> 🕮 *Un cas d'utilisation est une suite d'actions qui amène un résultat observable pour un acteur particulier. Des scénarios illustrent les cas d'utilisation par l'exemple.*

Ces cas d'utilisation sont regroupés dans des *paquetages* représentant les frontières du système.

> 🕮 *Un paquetage partitionne le domaine d'étude et les travaux associés. Il permet de regrouper divers éléments, en particulier des cas d'utilisations et des classes. Un paquetage peut aussi contenir d'autres paquetages. Les paquetages sont liés entre eux à travers des rapports contractuels définissant leur interface.*

Le paquetage est l'élément à partir duquel vous allez créer le diagramme de cas d'utilisation. Cependant, dans le cadre de systèmes complexes, vous pouvez également créer ce type de diagramme depuis un cas d'utilisation afin de détailler ce dernier.

Créer un paquetage

Pour créer un paquetage :

1. Ouvrez la fenêtre de navigation **Objets principaux**.
2. Cliquez avec le bouton droit sur le dossier **Paquetage** et sélectionnez **Nouveau** > **Paquetage**.
   La fenêtre de création d'un paquetage s'ouvre.
3. Saisissez son **Nom**.
4. Indiquez éventuellement la bibliothèque ou le paquetage détenteur.

   > ☛ *La bibliothèque par défaut est utilisée pour ranger un objet s'il n'y a pas de bibliothèque courante au moment de sa création.*
5. Cliquez sur **OK**.

Le paquetage est créé et ajouté dans la liste des paquetages.

> ☛ *Quand le bouton **OK** ou **Créer** est grisé, c'est que la fenêtre où il apparaît n'a pas été complètement renseignée.*

Créer le diagramme de cas d'utilisation du paquetage

Pour créer un diagramme de cas d'utilisation :

1. Cliquez avec le bouton droit sur le paquetage concerné.
2. Cliquez sur **Nouveau** > **Diagramme...**.
   Une fenêtre apparaît avec les types de diagrammes possibles.
3. Sélectionnez le type de diagramme **Diagramme de cas d'utilisation**.

4. Vérifiez que l'option **Initialiser le diagramme** est cochée.

   ☛ *L'option **Initialiser le diagramme** permet de prendre en compte l'environnement de l'objet dans le dessin.*

5. Cliquez sur **Créer**.
   Le diagramme créé s'ouvre dans la fenêtre d'édition de **MEGA**. Le cadre du paquetage est positionné à l'intérieur du dessin.

Ouvrir un diagramme de cas d'utilisation

Pour ouvrir un diagramme de cas d'utilisation existant :

1. Dans la fenêtre de navigation **Objets principaux**, faites un clic droit sur le paquetage qui contient le cas d'utilisation.
2. Cliquez sur **Diagramme de cas d'utilisation**.

Le diagramme de cas d'utilisation s'ouvre.

Vous êtes dans l'outil de dessin de **MEGA for UML**. Les paragraphes qui suivent présentent les objets d'un diagramme de cas d'utilisation : des acteurs, des cas d'utilisation, des paquetages, etc., ainsi que des liens entre ces objets.

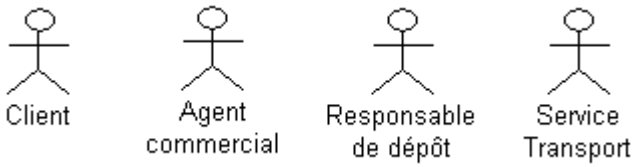
☛ *Dans les exemples présentés dans ce guide, les noms des objets incluent, entre autres, des blancs, des majuscules et des caractères accentués. Lorsqu'il y a, pour un outil de génération qui utilise les spécifications effectuées avec **MEGA for UML**, des restrictions sur les caractères autorisés et sur la longueur des noms, il est préférable de se conformer à ces règles dès la spécification avec **MEGA for UML**.*

LES ÉLÉMENTS DU DIAGRAMME DE CAS D'UTILISATION

Les acteurs

📖 *Un acteur représente le rôle joué par quelque chose ou quelqu'un se trouvant dans l'environnement de l'entreprise ou du système étudié. Il est en relation avec le métier de l'entreprise et interagit avec le système dans différents cas d'utilisation. Ce peut être un élément de la structure de l'entreprise tel qu'une direction, un service ou un poste de travail.*

Exemples d'acteurs :



Pour créer un *acteur* dans un diagramme de cas d'utilisation :

1. Dans la barre d'objets, cliquez sur le bouton **Acteur (UML)**  .
2. Cliquez sur le plan de travail du diagramme.
   La fenêtre **Ajout d'un acteur (UML)** s'ouvre.
3. Saisissez le nom de l'acteur, "Standardiste", par exemple.
4. Cliquez sur le bouton **Créer**.

   ☛ *Lorsque le nom que vous avez saisi correspond à un acteur qui existe déjà dans la base, le libellé du bouton **Créer** devient **Relier**.*

   L'acteur apparaît alors dans le diagramme.

   ☺ *Vous pouvez créer plusieurs éléments à la suite sans revenir à la barre d'outils en effectuant un double-clic sur le bouton* 

   ☺ *Pour revenir ensuite au mode normal, utiliser la touche <Echap>, ou cliquez sur un autre bouton de la barre d'outils, par exemple sur la flèche* 

   ☺ *Vous pouvez affecter la même taille à plusieurs éléments en leur appliquant la commande **Retailler à l'identique** du menu **Dessin**,*

*après les avoir sélectionnés. Vous pouvez de même les aligner à l'aide de la commande **Aligner** de ce même menu. (Le dernier élément sélectionné est pris comme référence. Pour sélectionner plusieurs éléments, cliquez successivement sur chaque élément en maintenant la touche <Majuscules> enfoncée.)*

Les cas d'utilisation

Exemples de *cas d'utilisation* : traitement d'une commande, livraison d'un client, ouverture d'un compte, envoi d'une facture, établissement d'un crédit, achat d'un billet d'avion, etc.

Pour créer un cas d'utilisation dans un diagramme :

1. Cliquez sur le bouton **Cas d'utilisation** dans la barre d'objets.
   La fenêtre **Ajout d'un cas d'utilisation** s'ouvre.
2. Saisissez le **Nom** du cas d'utilisation et cliquez sur le bouton **Créer**.

Le cas d'utilisation apparaît sur le diagramme.

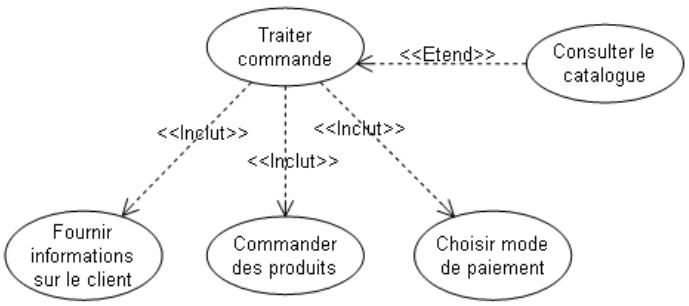
Faire un zoom sur un cas d'utilisation

Pour ouvrir directement, depuis le diagramme du paquetage, le diagramme qui décrit un cas d'utilisation :

1. Faites un clic droit sur le cas d'utilisation.
2. Sélectionnez **Diagramme de cas d'utilisation**.

> ☛ *Vous pouvez également créer un diagramme de classes ou un diagramme d'objets à l'aide de la commande Nouveau.*

Le diagramme de cas d'utilisation s'ouvre.

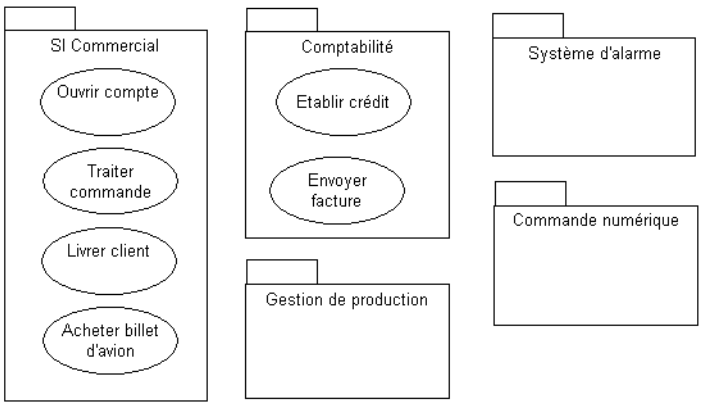


> ☺ *Ce zoom sur la description d'un objet est disponible sur tous les éléments décrits par un diagramme.*

Les paquetages

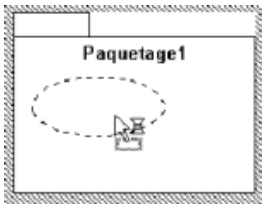
Un paquetage partitionne le domaine d'étude et les travaux associés. Il permet de regrouper divers éléments, en particulier des cas d'utilisations et des classes. Un paquetage peut aussi contenir d'autres paquetages. Les paquetages sont liés entre eux à travers des rapports contractuels définissant leur interface.

Exemples de *paquetage* : Le système d'information commercial, la comptabilité, la gestion de production, la commande numérique d'une machine, un système d'alarme, etc.



Vous pouvez créer un paquetage à l'aide du bouton  de la barre d'outils. Vous pouvez l'agrandir pour pouvoir poser des cas d'utilisation dessus.

Vous pouvez relier un cas d'utilisation à un paquetage simplement en le posant dessus. Lorsque vous déplacez l'objet sur le paquetage, la forme du paquetage est mise en relief pour indiquer que l'objet va bien lui être relié.



☛ *Si les objets reliés disparaissent sous le paquetage, cliquez sur le paquetage puis cliquez sur le bouton **Arrière-plan** de la barre de dessin.*

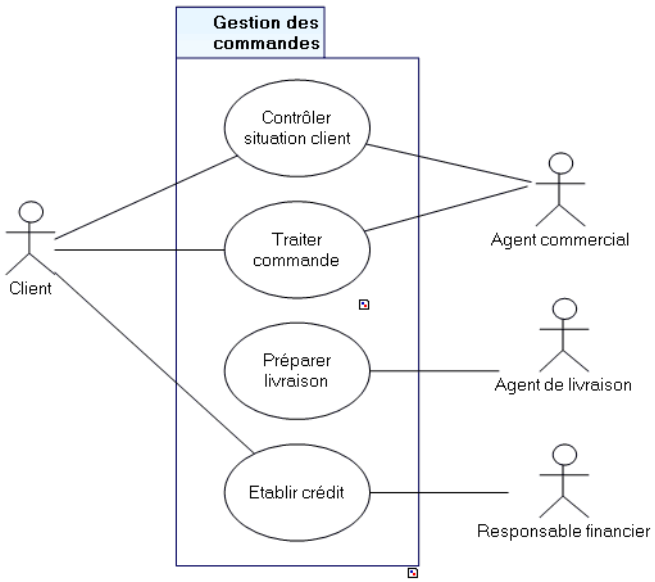
Lorsque vous déplacez un cas d'utilisation d'un paquetage à l'autre avec la souris, il est délié du premier et relié au deuxième. En revanche, si vous le déplacez avec les flèches du clavier, les liens ne sont pas modifiés.

Les participations

Vous pouvez indiquer quel acteur participe à chacun des différents cas d'utilisation.

Une participation indique qu'un acteur joue un rôle dans un cas d'utilisation.

Exemple de participation



- L'agent commercial intervient dans le traitement d'une commande et dans le contrôle de la situation du client ;
- L'agent de livraison intervient dans la livraison ;
- Le responsable financier intervient dans l'établissement des crédits, etc.

Créer une participation

Pour créer une *participation* dans le diagramme de cas d'utilisation :

1. Cliquez sur le bouton **Participation** de la barre d'objets.
2. Cliquez sur l'acteur concerné, et faites glisser la souris jusqu'au cas d'utilisation, avant de relâcher votre pression.
   Une fenêtre apparaît.
3. Indiquez le nom de la participation et indiquez si l'acteur en est l'initiateur.

*Il est possible de préciser le début du cas d'utilisation en activant la case à cocher **Initiateur** dans la fenêtre de propriétés de la participation correspondante. Une flèche apparaît sur le trait représentant la participation.*

4. Cliquez sur **OK**.

Le lien représentant la participation apparaît dans le diagramme.

Une participation est représentée par un lien mais il s'agit d'un objet, avec des propriétés qui lui sont propres.

☛ *La bobine n'est pas utilisée pour créer des participations. Elle sert à créer certains types de liens comme ceux entre Paquetage et les autres objets.*

☛ *En cas d'erreur, vous pouvez supprimer un objet en cliquant avec le bouton droit sur cet objet, et en sélectionnant la commande **Supprimer** dans le menu contextuel. Vous pouvez de même supprimer un lien en cliquant avec le bouton droit sur le lien, et en sélectionnant la commande **Supprimer** ou **Délier** dans le menu contextuel du lien.*

Multiplicités d'une participation

Sur une participation, il est possible de préciser la multiplicité :

- De l'acteur, afin d'indiquer que plusieurs instance de l'acteur sont impliquées dans une même instance du cas d'utilisation (exemple : les participants à une réunion).
- Du cas d'utilisation, afin d'indiquer qu'une même instance d'acteur intervient dans plusieurs instances du cas d'utilisation (exemple : un agent commercial qui traite plusieurs commandes d'un même client).

Pour définir les multiplicités sur une participation :

1. Faites un clic droit sur la participation et sélectionnez **Propriétés**.
2. Cliquez sur l'onglet **Caractéristiques**.
   Un premier cadre vous permet de définir la multiplicité de l'acteur, un second cadre permet de définir celle du cas d'utilisation.

Une fois définie, la multiplicité apparaît dans le diagramme.

Les associations entre cas d'utilisation : extension et inclusion

Lorsque la taille du système à décrire est importante, il est utile d'avoir des mécanismes de représentation adaptables au niveau de détail désiré. C'est ce que permettent les associations entre cas d'*utilisation*.

Lorsqu'un cas d'utilisation comprend trop de possibilités et d'exceptions, ces dernières sont représentées à part dans des extensions du cas d'utilisation standard.

Relation d'inclusion

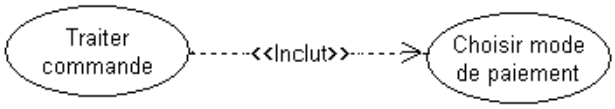
Un cas d'utilisation peut être sollicité automatiquement à la suite d'un autre, par exemple, la validation d'une commande inclut obligatoirement la sélection d'un moyen de paiement.

Pour indiquer qu'un cas d'utilisation en inclut un autre :

1. Dans le diagramme de cas d'utilisation, cliquez sur le bouton **Cas inclus**

2. Cliquez dans le cas utilisateur, par exemple "Traiter commande" et faites glisser la souris jusqu'au cas utilisé, par exemple, "Choisir mode de paiement", avant de relâcher votre pression.

Le lien apparaît alors dans le dessin accompagné du mot "Inclut".



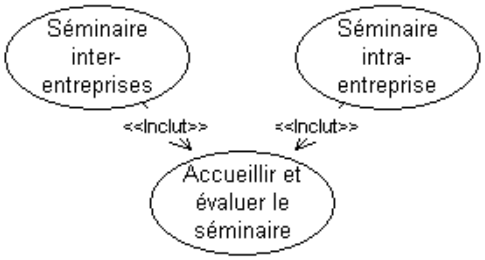
Exemples d'inclusion

Dans un organisme de formation, les cas d'utilisation :

- Séminaire inter-entreprises (dont les participants viennent de plusieurs entreprises différentes)
- Séminaire intra-entreprise (dont les participants viennent tous de la même entreprise)

peuvent avoir en commun le cas d'utilisation :

- Accueillir et évaluer le séminaire



Dans une entreprise commerciale, le cas d'utilisation :

- Passer une commande

peut réutiliser les cas d'utilisation suivants :

- Fournir des informations sur le client
- Passer un ordre de production
- Proposer un mode de paiement

Relation d'extension

Un cas d'utilisation peut entraîner l'exécution d'un autre. Contrairement à l'inclusion qui est automatique, l'extension est optionnelle.

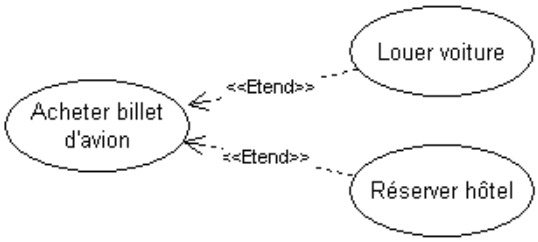
Pour indiquer qu'un cas d'utilisation est l'extension d'un autre :

1. Cliquez sur le bouton **Extension** 
2. Cliquez dans un cas d'utilisation, par exemple "Consulter le catalogue" et faites glisser la souris jusqu'au cas étendu, par exemple, "Traiter commande" avant de relâcher votre pression.
   La fenêtre de création d'une extension apparaît. Vous pouvez définir une contrainte ou un point d'extension.
3. Cliquez sur **Terminer**.

Le lien apparaît dans le diagramme accompagné du mot "Etend".

Exemple d'extension

L'achat d'un billet d'avion peut être complété par la réservation d'un hôtel ou la location d'une voiture.



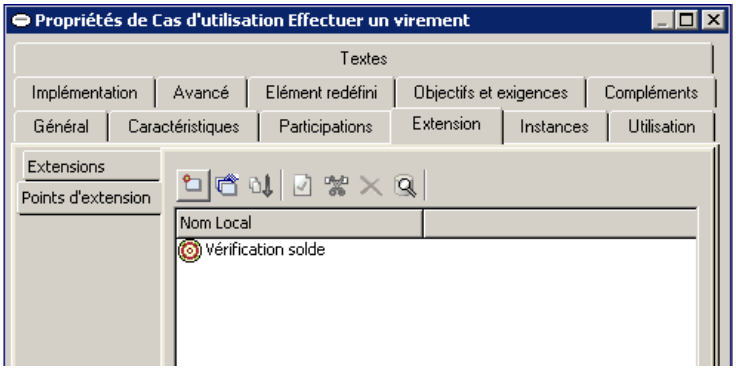
Point d'extension

L'extension peut intervenir à un point précis du cas étendu. Ce point est appelé point d'extension.

Pour créer un point d'extension sur le cas étendu :

1. Ouvrez la fenêtre de propriétés du cas d'utilisation étendu.
2. Cliquez sur l'onglet **Extension**, et le sous-onglet **Points d'extension**.

3. Cliquez sur le bouton **Nouveau**.
   Le point d'extension apparaît dans la fenêtre de propriétés. Vous pouvez le renommer.

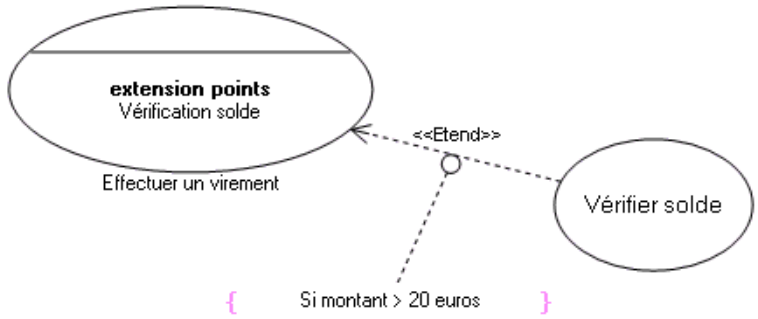


Un point d'extension peut être associé à une *contrainte* qui indique le moment où l'extension intervient. Vous pouvez ajouter une contrainte lors de la création de l'extension, ou ultérieurement, dans la fenêtre de propriétés du lien d'extension.

> 🕮 *Une contrainte représente un contrôle ou une règle de gestion qui doit être appliquée lors de l'exécution d'un traitement.*

Exemple de point d'extension

L'exemple suivant présente le cas d'utilisation d'un virement bancaire ; au delà de la somme de 20 euros, la vérification de solvabilité du client est déclenchée.



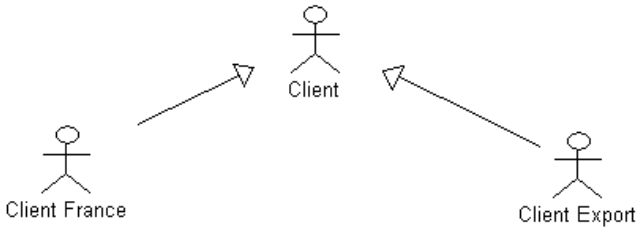
Les généralisations

> 🕮 *Une généralisation représente une relation d'héritage entre une classe générale et une classe plus spécifique. La classe spécifique est cohérente avec la classe plus générale et en hérite ses caractéristiques et son comportement. Elle comporte cependant des informations*

supplémentaires. Toute instance de la classe spécifique est aussi une instance de la classe générale.

La notion de *généralisation*, initialement utilisée pour les classes, a été étendue à d'autres concepts d'UML comme acteur et cas d'utilisation.

Exemples de généralisation entre acteurs :

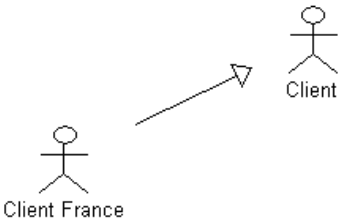


L'acteur "Client" peut être spécialisé en France et à l'Export.

Pour créer une généralisation entre acteurs :

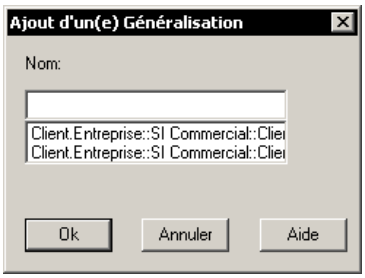
- Cliquez sur le bouton et tirez le lien en partant de l'acteur particulier (Ex : Client France) vers l'acteur général (Ex : Client).

La généralisation apparaît dans le dessin.



☛ *Vous pouvez créer de la même manière une généralisation entre deux cas d'utilisation.*

Lors de la création d'une deuxième généralisation, une fenêtre vous propose de réutiliser la généralisation existante si elle porte sur le même sujet.



Les interfaces

Il est possible de compléter la description d'un cas d'utilisation ou d'un acteur par la description des *interfaces* par lesquelles il communique avec son environnement.

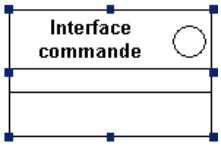
Pour créer une interface :

1. Cliquez sur le bouton **Interface** dans la barre d'objets.

   ☛ *Si le bouton **Interface** n'apparaît pas dans la barre d'objets, cliquez sur **Affichage > Vues et détails** et dans la fenêtre qui s'affiche, cochez la case **Classes**.*

2. Cliquez sur le plan de travail du diagramme.
3. Dans la fenêtre qui s'ouvre, saisissez le nom de l'interface et cliquez sur le bouton **Créer**.

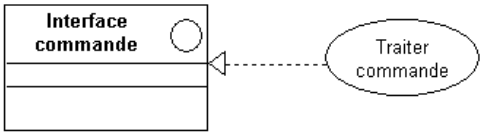
L'interface apparaît sur le diagramme.



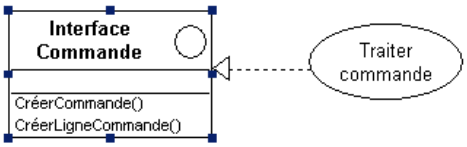
Pour préciser l'interface **supportée** par un cas d'utilisation :

- Cliquez sur le bouton et tirez le lien en partant du cas d'utilisation (Ex : Traiter Commande) vers l'interface (Ex : Interface Commande).

Le lien apparaît dans le dessin.



Vous pouvez préciser quelles sont les opérations que ce cas d'utilisation peut réaliser par l'intermédiaire de cette interface.



Pour indiquer l'interface **requise** par un cas d'utilisation :

- Cliquez sur le bouton  et tirez le lien entre le cas d'utilisation et l'interface.

✓ Pour plus de détails sur les interfaces requises et supportées, voir aussi "Relier les interfaces aux autres objets", page 96.

LE DIAGRAMME DE CLASSES

Un diagramme de classes permet de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et leurs relations. Un diagramme d'objets montre des exemples illustrant ce diagramme de classes.

La spécification des diagrammes de classes est souvent considérée comme la partie la plus importante dans la modélisation d'un système d'information. Les points abordés dans ce chapitre sont :

PRÉSENTATION DU DIAGRAMME DE CLASSES

Un diagramme de classes permet de représenter la structure statique d'un système, en particulier les types d'*objets* manipulés dans le système, leur structure interne et leurs relations.

> 🕮 *Un objet est une entité avec une identité et des frontières clairement définies dont l'état et le comportement sont encapsulés. Son état est défini par les valeurs de ses attributs et de ses liens avec d'autres objets. Son comportement est représenté par ses opérations et ses méthodes. Un objet est une instance de classe.*

Exemples d'objets :

- Objets de gestion :
  - Jacques Dupond, Pierre Durand, Paul Smith sont des instances de la classe "personne".
  - Les commandes no 10533 et 7322 sont des instances de la classe "commande".
  - Ecran SPD-1730 est une instance de la classe "article".
- Objets techniques utilisés pour la programmation :
  - Dlg_Order_Create, Dlg_Customer_Query sont des instances de la classe "fenêtre".
  - Str_Customer_Name, Str_Product_Comment sont des instances de la classe "chaîne".

Modéliser les données consiste à identifier les classes considérées d'intérêt pour représenter l'activité de l'entreprise, et à définir les associations qui existent entre elles.

Il faut que les classes et les associations qui constituent le diagramme de classes associé à un secteur de l'entreprise suffisent à le décrire complètement sur le plan sémantique.

En d'autres termes, on doit pouvoir décrire l'activité de l'entreprise en utilisant seulement ces classes et associations.

Ceci n'implique pas que, pour chaque mot ou verbe utilisé pour cette explication, il y ait un correspondant direct dans le diagramme de classes. Il s'agit de pouvoir traduire ce que l'on veut exprimer, au travers des classes et des associations.

La spécification des diagrammes de classes est souvent considérée comme la partie la plus importante dans la modélisation d'un système d'information.

Un diagramme d'objets montre des exemples illustrant ce diagramme de classes.

En particulier, il est possible d'illustrer un diagramme de classes par le diagramme d'objets correspondant dans un même dessin.

Le diagramme de classes : synthèse

Dans **HOPEX UML**, un diagramme de classes inclut :

- Des classes, qui représentent les concepts de base (client, compte, produit, etc.).
- Des associations, qui définissent les relations entre les différentes classes.
- Des attributs, qui décrivent les caractéristiques des classes et, dans certains cas, des associations.
- Des opérations, qui peuvent être effectuées sur les objets de la classe.

Le diagramme de classes est complété avec la définition des multiplicités.

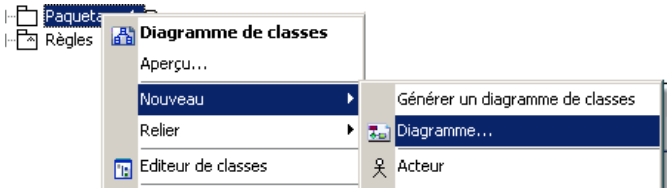
A la fin de ce guide, un glossaire rappelle la définition de chacun des concepts utilisés.

CRÉER UN DIAGRAMME DE CLASSES

Un diagramme de classe se crée à partir d'un paquetage.

Pour créer un diagramme de classes :

1. Faites un clic droit sur le nom du paquetage.
2. Dans le menu contextuel qui s'affiche, cliquez sur **Nouveau** > **Diagramme.**



☛ *Le menu contextuel peut être ouvert en cliquant sur un paquetage dans le navigateur ou dans un diagramme qui contient le paquetage.*

3. Dans la fenêtre qui apparaît, sélectionnez "Diagramme de classes" et cliquez sur le bouton **Créer**.

Le nouveau diagramme de classes s'ouvre.

Un diagramme de classes peut décrire un paquetage, un cas d'utilisation, une classe ou une instance.

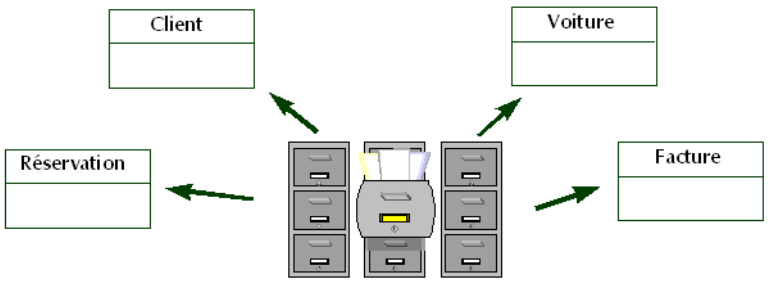
LES CLASSES

Définition d'une classe

Une *classe* est décrite par une liste d'attributs et d'opérations.

Une classe est reliée à d'autres classes via des *associations*. L'ensemble des classes et des associations constitue le noyau d'un diagramme de classes.

Nous pouvons illustrer la notion de classe en comparant les classes à des fiches dans des tiroirs.



Les classes peuvent être des objets techniques utilisés pour la programmation.

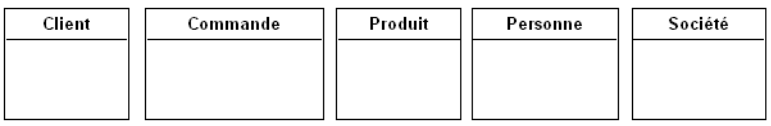
```
Exemples : fenêtre, rectangle, chaîne, tableau, etc.
```

Les classes peuvent représenter des objets techniques utilisés dans l'industrie.

```
Exemples : Alarme, Capteur, Zone
```

Les classes peuvent également représenter des objets de gestion.

```
Exemples : client, commande, produit, personne, société,
etc.
```



Une classe peut aussi être la traduction d'un processus comme "Valider demande client" ou l'implémentation d'une règle de gestion fonctionnelle comme "Cohérence comptes analytiques".

☛ *A la fin de ce guide, un glossaire rappelle la définition de chacun des concepts utilisés.*

Créer une classe

Pour créer une classe :

1. Cliquez sur le bouton **Classe**  de la barre d'insertion d'objets.
2. Puis cliquez sur le plan de travail du diagramme.
   La fenêtre **Ajout d'une classe** s'ouvre.
3. Saisissez le **Nom** de la classe et cliquez sur le bouton **Créer**.
   La classe est alors posée dans le diagramme.

   ☛ *Dans les exemples présentés dans ce guide, les noms des objets incluent, entre autres, des blancs, des majuscules et des caractères accentués. Il est important de noter que, lorsqu'il y a, pour un outil de génération qui utilise les spécifications effectuées avec **HOPEX UML**, des restrictions sur les caractères autorisés et sur la longueur des noms, il est préférable de se conformer à ces règles dès la spécification avec **HOPEX UML**.*

Vous pouvez créer plusieurs classes à la suite sans revenir à la barre d'outils en double-cliquant sur le bouton  .

Pour revenir ensuite au mode normal, cliquez sur la flèche 

Vous pouvez utiliser partout le nom complet d'une classe en indiquant le paquetage auquel elle appartient séparé par deux fois deux points.

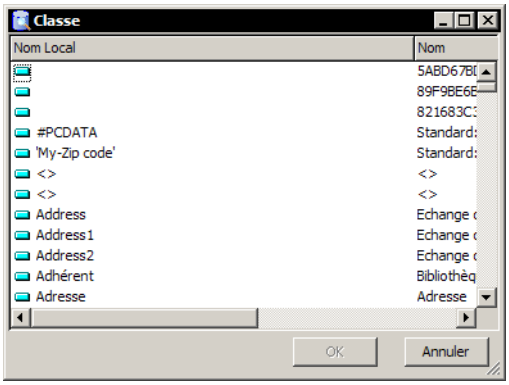
```
Exemple :
Entreprise::Gestion Commerciale::Client
```

Si l'un des paquetages cités n'existe pas, il est automatiquement créé et relié à la classe.

Retrouver une classe existante

Pour retrouver une classe existante:

1. Dans la fenêtre **Ajout d'une classe**, sélectionnez la commande **Lister** à l'aide de la flèche.
   La liste des classes apparaît.



2. Sélectionnez la classe qui vous intéresse et cliquez sur **OK**.
   Le nom de la classe sélectionnée apparaît dans la fenêtre d'ajout d'une classe.
3. Cliquez sur **Relier**.
   La classe apparaît dans le diagramme.

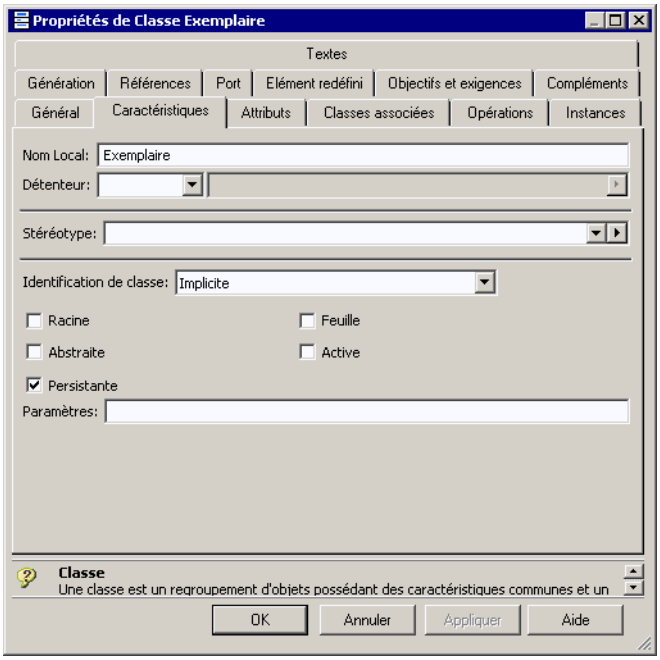
> ☛ *Notez que vous pouvez faire apparaître la liste des classes existantes en utilisant la touche <Ctrl-L>.*
>
> *Si la base contient de nombreuses classes, vous pouvez utiliser la fonction de recherche à l'aide de la touche <Ctrl-Q> pour effectuer une sélection plus fine. Voir le guide **HOPEX Power Studio** pour l'utilisation de cette fonctionnalité.*

Propriétés d'une classe

Les propriétés affichées varient en fonction du stéréotype de la classe.

Pour ouvrir la fenêtre de propriétés d'une classe :

- Cliquez avec le bouton droit sur la classe et sélectionnez **Propriétés**. Plusieurs onglets permettent de définir les propriétés d'une classe.



☛ *Le commentaire situé au bas de la fenêtre décrit le champ que vous avez sélectionné.*

Onglet Caractéristiques

L'onglet **Caractéristiques** permet de saisir différentes caractéristiques de la classe :

- Son **Nom**, que vous pouvez modifier.

  ☺ *Vous pouvez également modifier le nom d'une classe en cliquant directement sur son nom dans le dessin.*

- Le *paquetage* auquel appartient la classe.
- La **Visibilité** de la classe par rapport au paquetage :
  - "Publique" : la classe est visible par tout élément situé à l'extérieur du paquetage. C'est la visibilité sélectionnée par défaut.
  - "Protégée" : la classe est visible par les éléments héritiers ou importateurs et amis.
  - "Privée" : la classe est seulement visible par les éléments importateurs et amis.
  - "Non spécifiée".

    🕮 *Les amis d'une classe sont des classes autorisées à connaître les internes de cette classe. Il est possible de préciser les amis d'une classe dans l'onglet Compléments de la fenêtre de propriétés de la classe.*

- Son *stéréotype* : voir "Stéréotype d'une classe", page 36.

Les autres caractéristiques que vous pouvez préciser sont l'abstraction, la persistance et l'activité :

- Une classe **Abstraite** n'a pas d'instance. Elle n'est utilisée que pour mettre en commun des opérations ou des attributs communs à ses sous-classes.
- La **Persistance** précise si cette classe doit être conservée dans le temps ou si elle ne vit que le temps du traitement en mémoire dans l'ordinateur.
- Les instances d'une classe **Active** sont capables de déclencher des flux de contrôle sans qu'il y ait d'intervention de l'utilisateur.

  ```
  Ex : Une instance de la classe imprimante peut envoyer un
  message "Plus de papier disponible" sur l'écran de
  l'administrateur du réseau.
  ```

- Une classe **Racine** est une classe ne possédant pas de sur-classe dans l'arborescence des généralisations de classe.
- Une classe **Feuille** est une classe ne possédant pas de sous-classe dans l'arborescence des généralisations de classe.

Vous pouvez également préciser les Paramètres d'une classe paramétrée (pour C++).

☛ *Voir "Pour spécifier une classe paramétrée :", page 75 pour plus de détails.*

Onglet Général

Certaines propriétés générales apparaissent sur tous les éléments du diagramme.

- Le nom du créateur de l'élément.
- Le nom du dernier modificateur de l'élément.
- Le statut modifiable ou non de l'élément qui indique si vous avez le droit de modifier les caractéristiques de cet élément lorsqu'une gestion des autorisations est mise en place.

Onglet Textes

Le texte **Commentaire** vous permet de commenter une classe.

☺ *Les commentaires permettent de compléter le diagramme lorsque des détails utiles n'apparaissent pas dans le dessin. Ces commentaires sont repris dans le document qui décrit le diagramme de classes.*

Autres onglets

D'autres onglets vous permettent de définir ou de visualiser :

- Les attributs d'une classe (voir "Les attributs", page 38)
- Les opérations d'une classe (voir "Les opérations", page 43)
- Les classes associées (voir "Les associations", page 54)
- Les instances d'une classe (voir "Le diagramme d'objets", page 78)

Stéréotype d'une classe

Un stéréotype est un type d'élément de modélisation qui permet d'étendre la sémantique du métamodèle. Les stéréotypes doivent être basés sur des types ou des classes existantes dont ils reprennent la structure. D'autres stéréotypes peuvent être créés par l'utilisateur.

Les stéréotypes disponibles pour une classe sont :

- **Acteur** : représente le rôle joué par quelque chose ou quelqu'un se trouvant dans l'environnement du système étudié.
- **Auxiliaire** : classe qui supporte une autre classe centrale ou fondamentale, généralement en implémentant un flux logique ou de contrôle secondaire.
- **Classe d'implémentation** : permet de caractériser les classes qui sont nécessaires à l'implémentation physique du système.
- **Classe Meta** : classe dont les instances sont des classes. En règle générale, les méta-classes sont utilisées pour construire des méta-modèles.
- **Contrôle** : permet de caractériser les classes effectuant des traitements internes au système. Ceux-ci nécessitent généralement le concours de plusieurs classes.
- **Entité** : permet de caractériser des classes passives qui ne génèrent aucune interaction par elles-même. Elles peuvent participer à plusieurs cas d'utilisation et survivent généralement à une interaction unique. Elles représentent des objets partagés entre les différents acteurs qui les manipulent.
- **Enumération** : type de données contenant une liste de valeurs tabulées.
- **Expression** : expressions de types de données complexes basés sur des types.
- **Focus** : classe qui définit le flux logique ou de contrôle principal pour la ou les classes auxiliaires qui la supportent.
- **Frontière** : permet de caractériser les classes qui sont en prise directe avec l'environnement du système. Les interfaces hommes-machines en font partie.
- **Interface** : une interface est constituée d'un ensemble d'opérations qui décrivent le comportement d'un élément. En particulier, une interface représente la partie visible d'une classe ou d'un paquetage dans une relation contractuelle de type client - fournisseur.

  ☛ *Ce sont des interfaces entre les différents composants du système informatique. Ce ne sont pas des interfaces avec les utilisateurs du système qui sont du stéréotype frontière. Voir "Spécifier les interfaces", page 72 pour plus de détails.*

- **Opérateur** : représente un acteur humain qui interagit avec le système. Un opérateur interagit avec d'autres opérateurs et manipule des entités lorsqu'il participe à la réalisation d'un cas d'utilisation.
- **Opérateur externe** : un opérateur externe interagit directement avec des acteurs extérieurs au système.
- **Opérateur interne** : un opérateur interne interagit avec d'autres opérateurs et entités à l'intérieur du système.

- **PowerType** : méta-type dont les instances sont des sous-types d'un autre type.
- **Structure** : classe servant à décrire une structure utilisée dans les programmes.
- **Thread** : stéréotype utilisé dans l'implémentation d'un objet actif en tant que processus léger.
- **Type élémentaire** : permet de caractériser les types de données.
- **Utilitaire** : une classe de ce stéréotype regroupe des variables globales et des procédures utiles à la programmation décrites sous forme des attributs et opérations de cette classe.
- **Schema group** : classe décrivant un type d'élément XML, dont les sous-éléments forment un groupe.
- **XML Document Definition Root** : classe servant à décrire la structure d'un message échangé entre deux systèmes en utilisant la syntaxe du langage XML.

Option d'affichage des stéréotypes

Une option vous permet d'afficher les stéréotypes dans la fenêtre de navigation des objets.

Pour activer cette option :

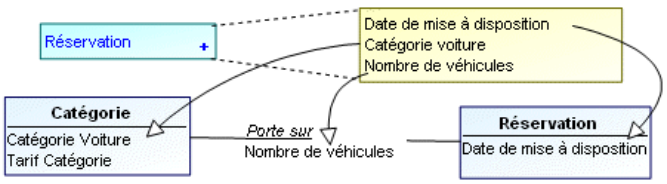
1. Depuis l'espace de travail **HOPEX**, cliquez sur le menu **Outils** > **Options**.
2. Dans la partie gauche de la fenêtre des options, sélectionnez le dossier **Espace de travail**.
3. Dans la partie droite, cochez l'option **Afficher le stéréotype des objets UML dans le navigateur**.
4. Cliquez sur **OK**.

LES ATTRIBUTS

Définition d'un attribut

Un attribut est une propriété nommée d'une classe. C'est la donnée élémentaire mémorisée dans le système d'information de l'entreprise.

```
Exemples :
"Nom du client" (Attribut de la classe client).
"No client" (Identifiant de la classe client).
"Solde du compte" (Attribut de la classe compte).
```

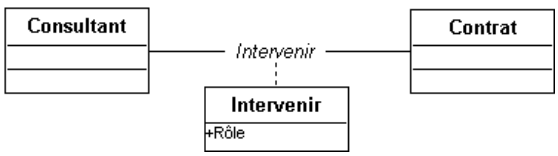


Les classes et les classes d'associations peuvent être caractérisées par des attributs.

Ces attributs ont pu, par exemple, être révélés par l'étude du contenu des messages qui circulent à l'intérieur de l'entreprise.

Un attribut est porté par une association quand sa valeur dépend de l'ensemble des classes participant à cette association.

Dans le diagramme présenté ci-après, le "rôle" qu'un "Consultant" a joué sur un "Contrat" dépend du consultant et du contrat, donc de l'association "Intervenir".



Spécifier les attributs d'une classe

Pour définir les attributs d'une classe :

1. Cliquez sur la classe avec le bouton droit de la souris.
2. Dans le menu contextuel, sélectionnez **Attributs**.

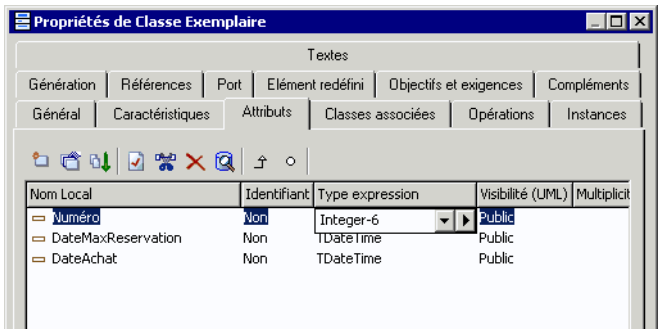
Les attributs de la classe y sont présentés. Il est possible de préciser pour chacun d'eux :

- Son **Type**, qu'il est possible d'exprimer sous forme d'une expression.

  `Ex : Integer.`

  ☛ *L'expression doit être conforme à la syntaxe UML. Voir "Signature d'une opération ou d'un signal", page 46 pour plus de détails.*

Pour consulter la liste des types possibles, cliquez dans le champ correspondant et déplacez-vous à l'aide de la flèche du menu déroulant.



Sont présents dans la liste les types qui appartiennent aux différentes classes du paquetage de départ (il s'agit ici de "Bibliothèque").

Pour rechercher d'autres types, cliquez sur la flèche ▶ à droite du menu déroulant puis sélectionnez **Rechercher**. La fenêtre **Rechercher** s'affiche.

Vous pouvez également taper directement dans le champ le nom du type de l'attribut.

- Sa **Visibilité** :
  - "Public": c'est la visibilité par défaut. L'attribut est visible par tous.
  - "Protégé": l'attribut est visible par les héritiers de son paquetage ou ses amis.
  - "Privé": l'attribut est visible par sa classe ou ses *amis*.

    🕮 *Les amis d'une classe sont des classes autorisées à connaître les internes de cette classe. Il est possible de préciser les amis d'une classe dans l'onglet Compléments de la fenêtre de propriétés de la classe.*
- Sa **Multiplicité**, c'est à dire le nombre de répétitions de cet attribut dans la classe.

Les boutons de la barre d'outils vous permettent d'effectuer les manipulations suivantes :


Créer un nouvel attribut ou plus généralement un nouvel élément.


Réordonner les éléments.


Afficher les propriétés d'un élément.


Supprimer un élément.


Explorer un élément.

Attributs hérités

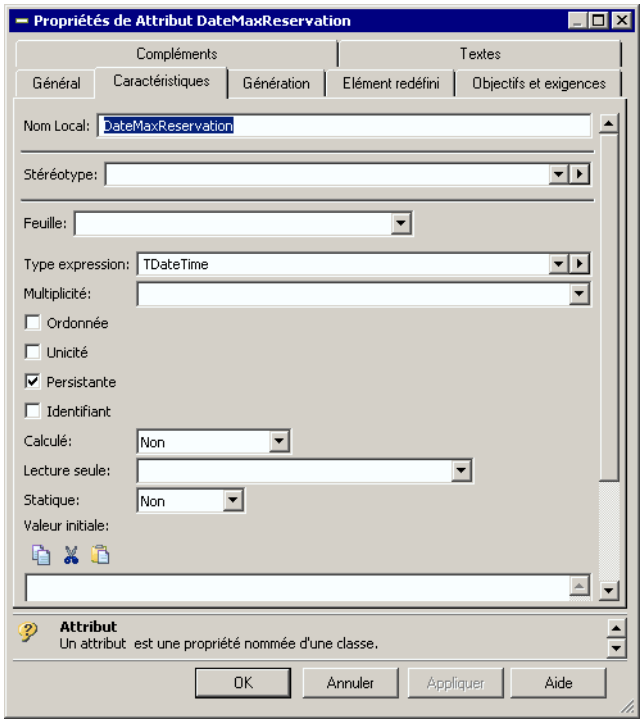
Lorsqu'une généralisation existe entre une classe générale et une classe particulière, la classe particulière hérite des attributs de la classe générale.

- Cliquez sur le bouton **Attributs hérités**  pour visualiser les attributs hérités d'autres classes.

Propriétés des attributs

Pour ouvrir la fenêtre de propriétés d'un attribut :

- Cliquez avec le bouton droit sur l'attribut et sélectionnez **Propriétés**.



Dans cette fenêtre, vous pouvez préciser :

- Le **Type** de l'attribut sous la forme d'une **expression** (voir "Types", page 42).
- S'il s'agit d'un attribut **statique** : indique si l'attribut peut prendre des valeurs spécifiques pour chacune des instances de la classe ou bien avoir une valeur qui caractérise l'ensemble de la classe.
  - "Oui" : l'attribut a une valeur qui caractérise l'ensemble de la classe. Par exemple, l'attribut "Longueur des numéros de téléphone" de la classe "Client France" est de 10 chiffres.
  - "Non" : l'attribut peut prendre une valeur différente pour chacune des instances de la classe. Par exemple, l'attribut "Numéro de téléphone" prend une valeur différente pour chaque instance de la classe "Client".
- S'il s'agit d'un attribut **Persistant**, c'est-à-dire si la valeur de cet attribut doit être conservée dans le temps ou si elle ne vit que le temps du traitement en mémoire dans l'ordinateur.
- Sa **Multiplicité**, c'est-à-dire le nombre de répétitions de cet attribut dans la classe.
- S'il est en **Lecture seule**, c'est-à-dire si sa valeur peut être modifiée après avoir été renseignée une première fois.

- S'il s'agit d'un **Attribut Calculé**, c'est-à-dire que sa valeur est déduite de la valeur d'un ou plusieurs autres attributs.
- La **Valeur initiale** de l'attribut, qu'il prendra lors de la création d'une instance de la classe.
- Sa **Visibilité** :
  - "Public" : c'est la visibilité par défaut. L'attribut est visible par tous.
  - "Protégé" : l'attribut est visible par les héritiers de son paquetage ou ses amis.
  - "Privé" : l'attribut est visible par sa classe ou ses amis.

Types

Un type permet de mettre en commun des caractéristiques communes à plusieurs attributs. Un type est implémenté sous forme de classe.

Toute classe peut être utilisée pour typer un attribut ou un paramètre.

```
Exemple : Client, Commande, Fenêtre, Tableau
```

Les classes de stéréotype "Type élémentaire" sont créées uniquement afin de typer des attributs ou des paramètres. Elles sont invariables.

```
Exemples de types élémentaires :

Chaîne.

Entier.

Adresse Export.

Montant en devises.
```

Vous pouvez lister les types existants à l'aide de la flèche  ou en créer de nouveaux à l'aide de la flèche 

Les types proposés sont les classes détenues ou utilisées par le paquetage ou par les paquetages dont il est client.

> ☛ *Pour renseigner la structure d'un type, posez la classe correspondante dans le même diagramme ou dans un autre diagramme et activez la commande Propriétés de son menu contextuel.*

Des informations complémentaires sur l'utilisation des types élémentaires sont disponibles dans l'annexe de ce guide.

LES OPÉRATIONS

Définition d'une opération

Une opération est un service qui peut être demandé à un objet pour mettre en oeuvre un comportement défini. Une opération possède une signature qui permet de préciser les paramètres qui lui sont nécessaires.

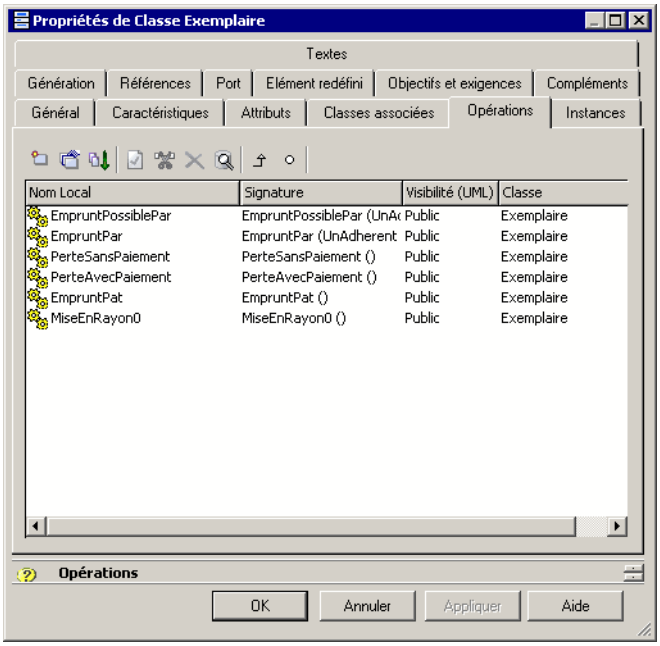
```
Exemples :
"Calcul Age" (Opération de la classe client).
"Imprimer" (Opération de la classe dessin).
"Calculer échéancier" (Opération de la classe prêt).
```

Spécifier les opérations d'une classe

Pour spécifier les opérations d'une classe :

➢ Cliquez avec le bouton droit sur la classe et sélectionnez **Opérations**.

Une fenêtre présentant les opérations de la classe s'ouvre.



Vous pouvez ajouter des opérations en cliquant sur le bouton  et préciser leur signature.

Opérations héritées

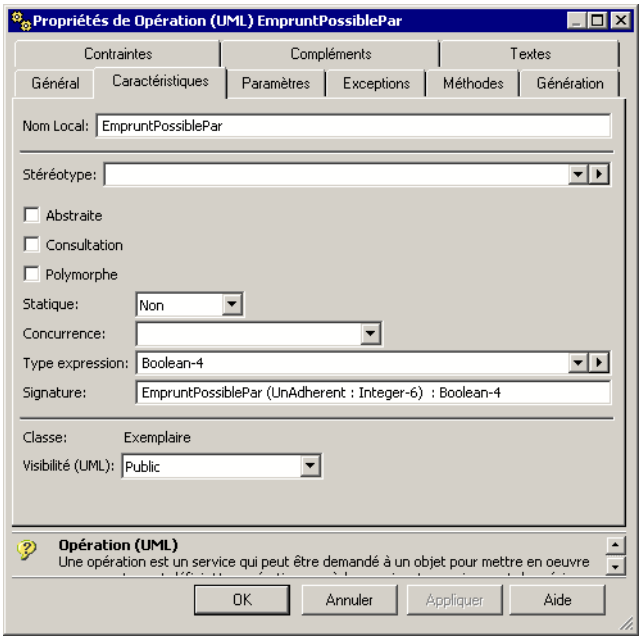
Lorsqu'une généralisation existe entre une classe générale et une classe particulière, la classe particulière hérite des opérations de la classe générale.

- Cliquez sur le bouton **Opérations héritées**  pour visualiser les opérations héritées d'autres classes.

Propriétés d'une opération

Pour ouvrir la fenêtre de propriétés d'une opération :

- Cliquez avec le bouton droit sur l'opération et sélectionnez **Propriétés**.



Vous pouvez indiquer pour chaque opération :

- Son **Stéréotype** afin de préciser son utilisation :
  - **Constructeur** : créer une instance de la classe.
  - **Destructeur** : détruire une instance de la classe.
  - **Itérateur** : parcourir les instances de la classe.
  - **Sélecteur** : effectuer une sélection parmi les instances de la classe.
- S'il s'agit d'une opération **Statique** : si l'opération qui peut prendre des valeurs spécifiques pour chacune des Instances de la classe ou bien avoir une valeur qui caractérise l'ensemble de la Classe.
- La **Concurrence** permet de préciser le comportement de l'opération lorsqu'elle est appelée plusieurs fois simultanément.
  - **Concurrente** : l'opération répond simultanément aux différents appels.
  - **Protégée** : l'opération répond au premier appel et rejette les suivants.
  - **Séquentielle** : l'opération répond successivement à chacun des appels.
- Si c'est une opération en **Consultation**, c'est-à-dire qui ne modifie pas l'état de l'objet.
- Si cette opération est **Polymorphe**, c'est-à-dire que des méthodes peuvent être redéfinies pour cette opération dans des sous-classes.
- Sa **Visibilité** :

- **Publique** : c'est la visibilité par défaut. L'opération est visible par tous.
- **Protégée** : l'opération est visible par les héritiers de son paquetage ou ses amis.
- **Privée** : l'opération est visible par sa classe ou ses amis.

Les indications suivantes sont utilisées pour compléter la signature de l'opération.

- Le **Type expression** de l'opération.

  Le type expression d'une opération précise le type de la variable retournée par l'opération à la fin de son exécution.

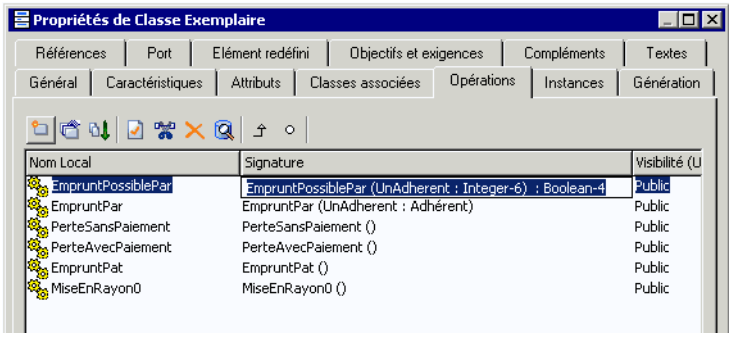
- Sa **Signature**.

Signature d'une opération ou d'un signal

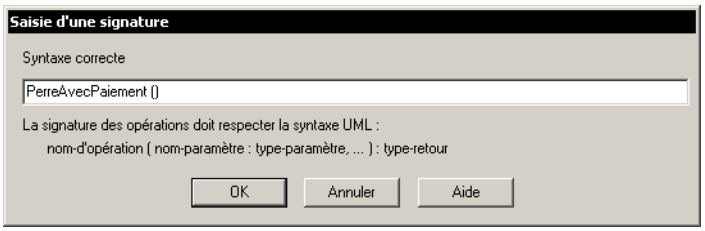
La signature d'une opération ou d'un signal est constituée du nom de l'opération (du signal), de son type de retour, et de ses paramètres avec leurs types. La syntaxe UML standard est utilisée pour cette signature, elle est du type : Ope0 (Param0 : M-Bool) : M-Bool.

Il est possible de définir la signature :

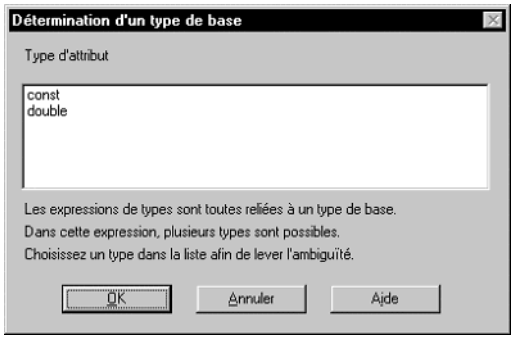
- Soit dans la fenêtre de propriétés de l'opération ou du signal.
- Soit dans la fenêtre de propriétés de la classe à laquelle appartient l'opération, en saisissant directement la **Signature** dans l'onglet **Opérations**.



Lorsque la signature est directement saisie, un contrôle de sa validité est effectué. En cas d'erreur, une fenêtre précise la nature de l'erreur et permet de la corriger.



De même, lorsque la signature comprend un type suivi ou précédé de complément (par exemple, Ope1(param1: const double), il est nécessaire de lever l'ambiguïté, en choisissant parmi un des mots composant l'expression.



Si le type n'existe pas, il est automatiquement créé.

> ☛ *Il est possible de filtrer les mots qui ne correspondent jamais à un type en créant un objet de type "_UML ReservedWord" et en lui donnant comme nom le mot à filtrer (en métamodèle avancé :Outils > Options : Référentiel).*

La signature qui est conservée inclut une référence au type : si le type est renommé, les signatures qui l'utilisent reflètent cette évolution.

Syntaxe des signatures

La syntaxe standard des signatures est :

```
nomopération(paramètre1:expressiondutype1,paramètre2:expres
siondutype2,...):expressiondutyperetour
```

Les noms comportant des blancs ou des caractères spéciaux doivent être placés entre apostrophes ('Nom du client'). Lorsqu'un nom contient une apostrophe, il faut dupliquer l'apostrophe : 'Nom de l' 'acheteur'

```
Exemples de signatures :

Déstocker (Produit0 : Entier(3), Quantité0 : Entier) :
Booléen

'Création de commande' ('Nom du client' : Client) : Byref
Variable
```

Dans la spécification d'une signature, il est possible de préciser le paquetage auquel appartient une classe, séparé par deux fois deux points.

```
Exemple : Entreprise::'Gestion Commerciale'::Client.
```

La classe citée est reliée au paramètre ou au type de retour. Si elle n'existe pas, elle est créée. De même, les paquetages cités dans le parcours qui n'existent pas sont créés et reliés à la classe.

Si le paquetage n'est pas précisé, une fenêtre vous propose de choisir entre les éventuelles classes homonymes.

Paramètres d'une opération

Dans la fenêtre de propriétés d'une opération, l'onglet **Paramètres** vous permet de préciser :

- Son **Type** sous forme d'une expression Ex : Integer(5).
- Sa **Valeur par défaut** Ex : 0.
- Sa **Direction** : en entrée et/ou en sortie de l'opération.

Pour créer un nouveau *paramètre* :

- Faites un clic droit dans la fenêtre et sélectionnez **Nouveau**.

> 🕮 *Un paramètre est la spécification d'une variable qui peut être modifiée, transmise ou retournée. Un paramètre peut préciser un nom, un type et une direction. Les paramètres sont utilisés pour les opérations, les messages et les événements.*
>
> 🕮 *Un argument est une valeur spécifique correspondant à un paramètre.*

Méthodes d'une opération (comportement opaque)

Une méthode - ou comportement opaque - est une représentation textuelle de l'implémentation d'une opération, d'une classe ou d'un composant. Elle spécifie l'algorithme ou la procédure qui produit les résultats d'une opération ou le comportement d'un élément.

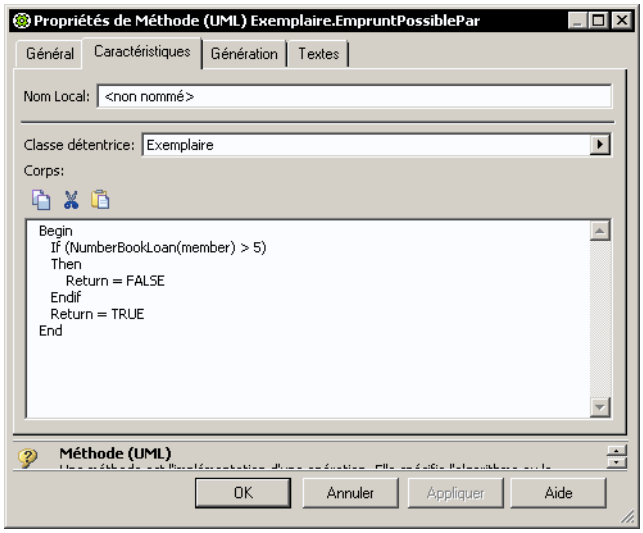
Pour définir la méthode implémentant une opération :

1. Ouvrez la fenêtre de propriétés de l'opération.
2. Sous l'onglet **Méthode**, cliquez sur le bouton **Nouveau**.
3. Cliquez dans la colonne **Classe détentrice** et sélectionnez **Classes candidates (Opérations)**.
4. L'outil recherche les classes susceptibles de fournir les méthodes d'implémentation.
   Vous pouvez également créer une nouvelle classe détentrice de la méthode.

Pour saisir le corps du texte de la méthode qui implémente l'opération :

1. Ouvrez la fenêtre de propriétés de la méthode.
2. Cliquez le sous-onglet **Caractéristiques**.

3. Définissez la méthode dans le cadre **Corps**.



Quand une classe possède plusieurs sous-classes, chaque sous-classe est susceptible d'implémenter l'opération sous forme d'une méthode différente.

L'onglet **Méthode** présente la méthode relative à la classe sélectionnée.

Conditions d'une opération

Vous pouvez définir les conditions d'une opération sous la forme de contraintes.

L'onglet **Contraintes** de la fenêtre de propriétés de l'opération vous permet de spécifier :

- La **Pré-Condition**, qui doit être vérifiée avant que l'opération ne s'exécute.
- Le **Corps**, qui doit être vérifié lors de l'exécution de l'opération.
- La **Post-Condition** qui doit être vérifiée après l'exécution de l'opération.

Si une condition n'est pas respectée, une exception est générée.

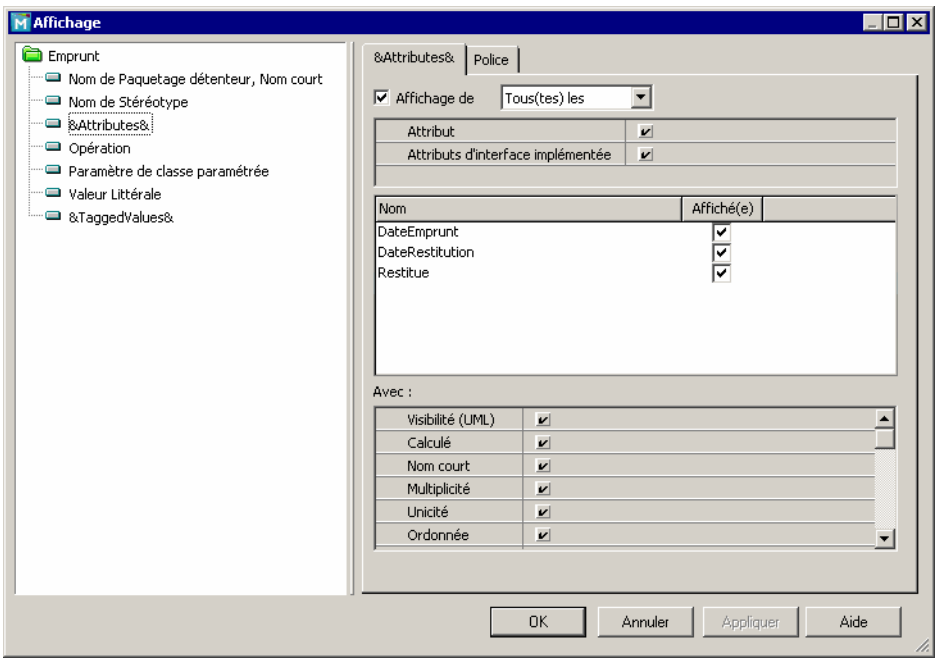
Exceptions d'une opération

L'onglet **Exceptions** vous permet de présenter les messages d'erreur envoyés par l'opération lorsqu'une exception se produit et de préciser leur signature.

Afficher les attributs et les opérations d'une classe

Pour modifier l'affichage des attributs et des opérations de la classe :

1. Cliquez avec le bouton droit sur la ou les classes dont vous voulez afficher des attributs.
2. Sélectionnez **Formes et détails**.
   La fenêtre de sélection des éléments à afficher apparaît.
3. Choisissez **Attribut** dans l'arbre qui vous est présenté.



4. Sélectionnez les attributs que vous voulez afficher.

Vous pouvez les afficher **Tous**, en afficher **Certains** que vous allez sélectionner dans la liste, ou n'en afficher **Aucun**.

Vous pouvez demander l'affichage de la **Visibilité**, du *type*, … de chacun des attributs.

> 🕮 *Un type de données permet de mettre en commun des caractéristiques communes à plusieurs attributs. Les types de données sont implémentés sous forme de classe.*
>
> ☺ *Vous pouvez cacher ou rendre visible le compartiment présentant les attributs dans le dessin de la classe, en activant ou désactivant la case à cocher devant "Affichage de"*

L'affichage des opérations est indiqué de la même manière, en choisissant **Opération** dans l'arbre qui vous est présenté.

LES SIGNAUX

Définition d'un signal

Un signal est un événement qui peut être invoqué explicitement. Un signal peut posséder des paramètres. Un signal peut être envoyé à un objet ou à un ensemble d'objets. Il peut être invoqué dans le cadre de la participation d'un acteur à un cas d'utilisation.

Un signal peut être émis ou reçu par une classe. Il peut être également émis par une opération à la suite d'une exception.

Spécifier les signaux d'une classe

Créer un signal émis ou reçu

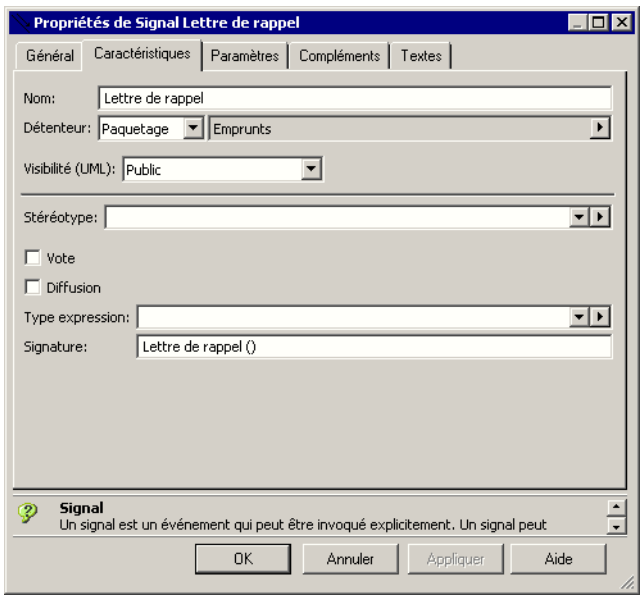
Pour préciser quels sont les signaux qui peuvent être émis ou reçus par une classe :

1. Ouvrez la fenêtre de propriétés d'une classe.
2. Cliquez sur l'onglet **Compléments**.
3. Dans l'arbre qui vous est présenté, sélectionnez la branche **Signal émis** ou **Signal reçu**, puis cliquez sur le bouton 
4. Indiquez le nom du signal et cliquez sur **OK**.

Propriétés d'un signal

Pour ouvrir la fenêtre de propriétés d'un signal :

1. Dans la fenêtre de propriétés d'une classe, dans l'onglet **Compléments**, cliquez sur le signal avec le bouton droit de la souris et sélectionnez **Propriétés**.
   La fenêtre de propriétés du signal apparaît.



Vous pouvez indiquer pour un signal :

- Son **Stéréotype** afin de préciser son utilisation :
  - **Exception** : signal d'erreur généré lorsqu'une exception se produit pendant l'exécution d'une opération.
- Sa **Visibilité** par rapport au paquetage :
  - **Publique** : c'est la visibilité par défaut. Le signal est visible par tout élément situé à l'extérieur du paquetage.
  - **Protégée** : le signal est visible par les éléments héritiers ou amis.
  - **Privée** : le signal est visible par sa classe ou ses amis.

    Les amis d'une classe sont des classes autorisées à connaître les internes de cette classe. Il est possible de préciser les amis d'une classe dans l'onglet Compléments de la fenêtre de propriétés de la classe.
- Le **Type expression** du signal (voir *type expression*).

  Le type expression d'un signal précise le type de la variable retournée par le signal lors de sa réception par son destinataire.

Un signal peut être une demande de **Vote** envoyé à chacun des objets actifs pour leur demander s'il est possible d'effectuer une action, comme par exemple fermer une session Windows.

Un signal peut donner lieu à une **Diffusion** générale à tous les objets actifs.

Paramètres d'un signal

Les *Paramètres* du signal sont renseignés sous l'onglet **Paramètres** de sa fenêtre de propriétés. Vous pouvez préciser :

- Son **Type** sous forme d'une expression. Ex : Integer(5).
- Sa **Valeur** par défaut. Ex : 0.
- Sa **Direction** : en entrée et/ou en sortie de l'opération.

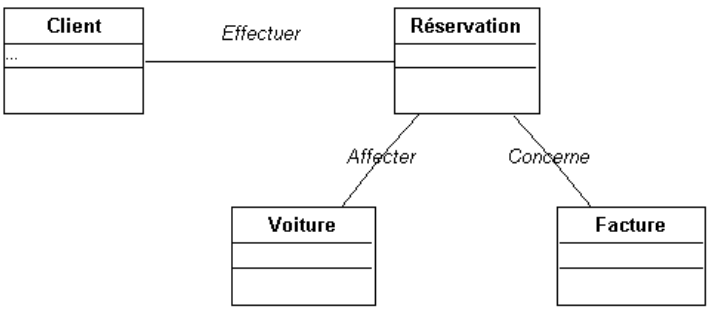
> 🕮 *Un paramètre est la spécification d'une variable qui peut être modifiée, transmise ou retournée. Un paramètre peut préciser un nom, un type et une direction. Les paramètres sont utilisés pour les opérations, les messages et les événements.*
>
> 🕮 *Un argument est une valeur spécifique correspondant à un paramètre.*

LES ASSOCIATIONS

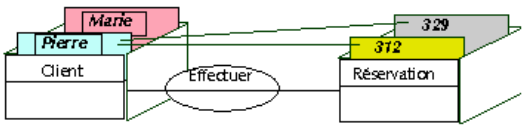
Une association est une relation existant entre deux classes.

Une association est binaire quand elle relie deux classes, ternaire quand elle en relie trois, etc.

Les associations peuvent être comparées à des liens entre des fiches.



Le dessin suivant permet de visualiser "en trois dimensions" les situations qu'un diagramme de classes permet de mémoriser.



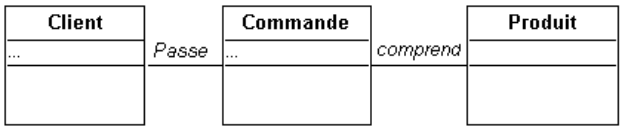
Pierre et Marie sont des clients. Pierre a effectué les réservations numéros 312 et 329.

Un diagramme de classes doit permettre de mémoriser toutes les situations du contexte de l'entreprise.

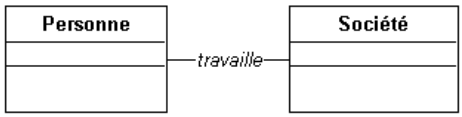
> ☛ *Le diagramme ne doit pas permettre de représenter des situations irréalistes ou aberrantes.*

Exemples d'association

- Un client *passe* une commande.
- Une commande *comprend* plusieurs produits.



- Une personne *travaille* pour une société.



- Une alarme *est déclenchée* par un capteur.

Un capteur *couvre* une zone.

Une fenêtre *affiche* une chaîne de caractères.

Créer une association

Pour créer une *association*:

1. Cliquez sur le bouton **Association**  de la barre d'objets.
2. Cliquez dans une des classes concernées, et faites glisser la souris jusqu'à la deuxième classe avant de relâcher votre pression. L'association apparaît dans le diagramme sous forme d'un trait.

S'il existe déjà une association entre les deux classes, la fenêtre **Ajout d'une association** s'ouvre. Elle permet de créer une nouvelle association ou de choisir entre les associations existantes.

Vous pouvez préciser un nom pour l'association dans sa fenêtre de propriétés, accessible par son menu contextuel.

> ☛ *En cas d'erreur, vous pouvez supprimer un élément ou un lien en cliquant avec le bouton droit sur cet élément, et en sélectionnant la commande Supprimer dans son menu contextuel.*

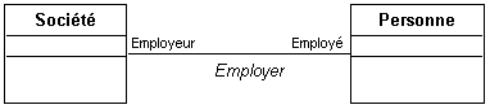
Vous pouvez également créer une association à partir de l'éditeur de classes. Voir "Créer des objets dans l'éditeur de classes", page 85.

Les rôles des associations

Il est possible de décrire les différents *rôles* joués par les classes dans les associations, et de préciser leur multiplicité et leur navigabilité.

Chaque extrémité d'une association permet de préciser le rôle joué par chaque classe dans l'association.

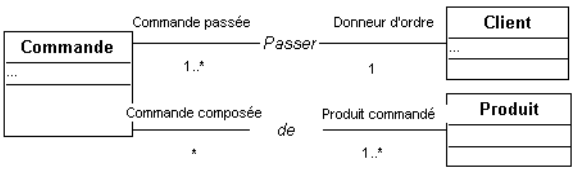
Visuellement, le nom du rôle se distingue du nom d'une association, car il est placé près de son extrémité. De plus, il apparaît en caractères droits, alors que le nom de l'association est en italique.



Lorsque deux classes sont reliées par une seule association, le nom des classes suffit souvent à caractériser le rôle ; nommer les rôles prend tout son intérêt lorsque plusieurs associations relient deux classes.

Exemples de rôles :

- Un client est le *donneur d'ordre* d'une commande.
- Une commande est *passée* par un client.
- Une commande est *composé*e de produits.
- Un produit est *commandé*.



Une personne est un *employé* dans une société.

Une société est *l'employeur* de ces personnes.

Une alarme est *déclenchée* par un ou plusieurs capteurs.

Une zone est *couverte* par un capteur.

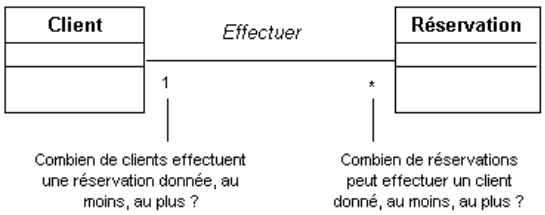
Une ou plusieurs chaînes de caractères sont *affichées* dans une fenêtre.

Multiplicité d'un rôle

*La multiplicité précise l'intervalle entre les valeurs minimum et maximum des cardinalités possibles pour un ensemble. On l'indique en particulier pour chacun des rôles que jouent les classes dans une association. Elle peut prendre les valeurs *, 0..1, 1, 1..*, 2..*, 4..10, etc. La valeur proposée par défaut est *.*

La cardinalité est le nombre d'éléments contenus dans un ensemble.

La *multiplicité* exprime le nombre minimum et maximum d'instances d'une classe pouvant être reliées par l'association à chaque instance de l'autre classe.

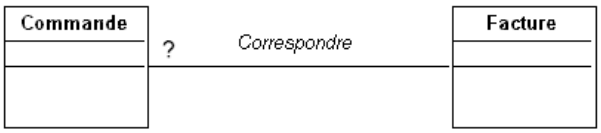


Les multiplicités usuelles sont "1", "0..1", "*" ou "0..*", "1..*", et "M..N" où "M" et "N" sont des entiers :

- La multiplicité "1" indique qu'une et une seule instance de la classe est reliée par cette association à chaque instance de l'autre classe.
- La multiplicité "0..1" indique qu'au plus une instance de la classe peut être reliée par cette association à chaque instance de l'autre classe.
- La multiplicité "*" ou "0..*" indique qu'un nombre quelconque d'instances de la classe peuvent être reliées par l'association à chaque instance de l'autre classe.
- La multiplicité "1..*" indique qu'au moins une instance de la classe est reliée par l'association à chaque instance de l'autre classe.
- La multiplicité "M..N" indique qu'au moins M instances de la classe et au plus N sont reliées par l'association à chaque instance de l'autre classe.

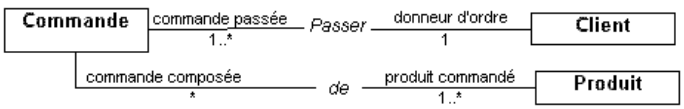
1	Un et un seul
0..1	Zéro ou un
M..N	De M à N (entiers naturels)
*	De zéro à plusieurs
0..*	De zéro à plusieurs
1..*	De un à plusieurs

L'exemple suivant va nous permettre d'illustrer la signification de chacune des multiplicités.



- 0..1 : A une commande correspond une facture au maximum ou aucune.
- * : Aucune restriction n'est imposée sur le nombre de factures correspondant à une commande.
- 1 : A chaque commande correspond une facture et une seule.
- 1..* : A chaque commande correspond une ou plusieurs factures.

Autres exemples de multiplicité :

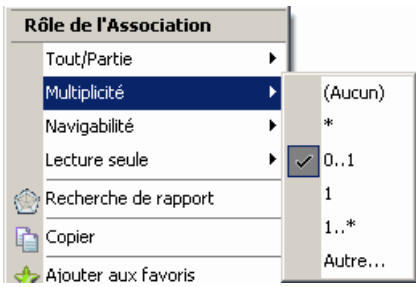


- 1..* : Un client peut passer une ou plusieurs commandes.
- 1 : Une commande est passée par un et un seul client.
- 1..* : Une commande comprend un ou plusieurs produits.
- * : Un produit peut faire partie de plusieurs commandes, ou d'aucune.
- 0..1 : Une personne travaille pour une société.
- 1..* : Une alarme est déclenchée par un ou plusieurs capteurs.
- 1 : Un capteur couvre une et une seule zone.
- 1..* : Une fenêtre affiche une ou plusieurs chaînes de caractères.

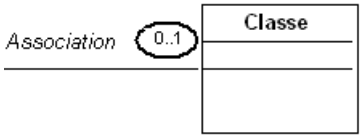
Préciser la multiplicité d'un rôle

Pour préciser la multiplicité d'un rôle :

1. Cliquez avec le bouton droit sur le trait qui se trouve entre l'association et la classe, et sélectionnez **Multiplicité**.



La multiplicité apparaît alors sur le rôle.



☛ *Si le menu affiché ne propose pas les multiplicités, contrôlez que vous avez bien cliqué sur le trait qui matérialise le rôle, et non sur l'association.*

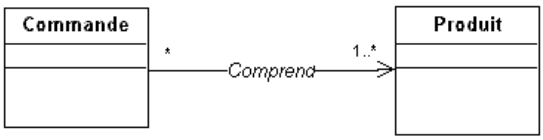
☺ *Toutes les informations précisées avec le menu contextuel peuvent être consultées et modifiées dans la fenêtre de propriétés du rôle.*

Navigabilité d'un rôle

La navigabilité précise le sens dans lequel l'association entre deux classes peut être parcourue. Pour ne pas encombrer le dessin, on n'indiquera la navigabilité que lorsqu'elle n'a lieu que dans un seul sens.

Exemple de navigabilité :

- Il est nécessaire de connaître tous les produits contenus dans une commande.
- Par contre, il est rarement utile de connaître toutes les commandes qui portent sur un produit.



Préciser la navigabilité d'un rôle

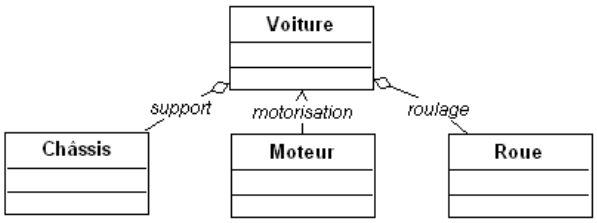
Pour indiquer qu'une association n'est navigable que dans un sens :

1. Cliquez avec le bouton droit sur le rôle non navigable.
2. Sélectionnez **Navigabilité > Non**.
   Une flèche représentant la navigabilité apparaît alors sur le rôle opposé.

Agrégation d'un rôle

L'agrégation est une forme particulière d'association qui indique que l'une des classes contient l'autre.

Exemple : Une voiture comprend un châssis, un moteur et des roues.



Préciser l'agrégation d'un rôle

Pour préciser l'agrégation d'un rôle :

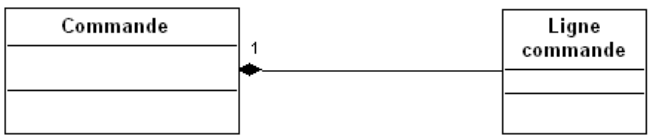
1. Cliquez avec le bouton droit sur le rôle.
2. Sélectionnez **Tout/Partie** > **Agrégat**.
   Un losange représentant l'agrégation apparaît alors sur le rôle.

Composition d'un rôle

La composition est une agrégation forte pour laquelle la durée de vie des composants coïncide avec celle du composé. Une composition est une agrégation immuable avec une multiplicité 1.

Exemple : Une commande est composée de plusieurs lignes de commande qui n'existent plus si la commande est supprimée.

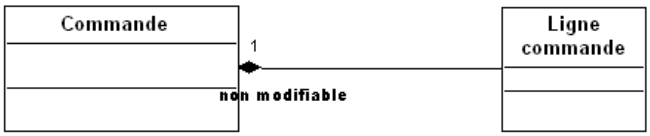
La composition est matérialisée par un losange noir.



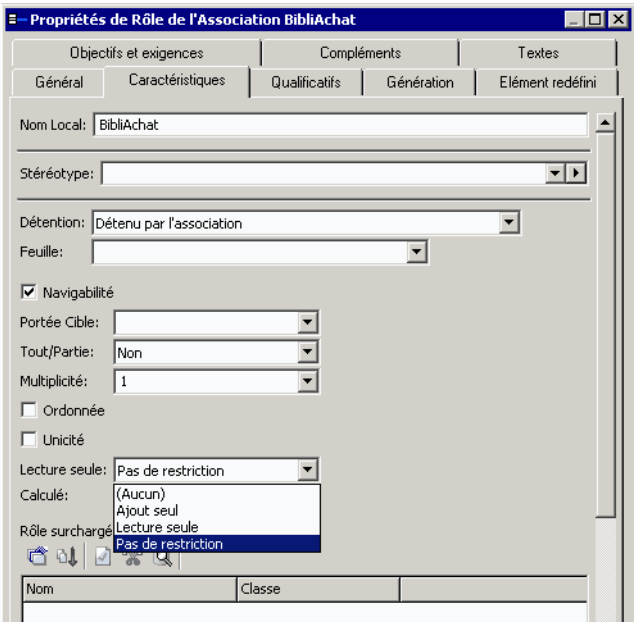
Rôle modifiable

La caractéristique **Lecture seule** permet de préciser si le rôle joué par une classe dans une association est modifiable après qu'il a été créé ou non. Par défaut, le rôle d'une classe dans une association est considéré comme modifiable.

```
Exemple : Une commande comprend une ligne de commande pour
chacun des produits commandés. On ne peut plus changer ces
différentes lignes de commande après l'enregistrement de la
commande.
```



Il est possible d'indiquer si un rôle est modifiable à l'aide du menu contextuel du rôle ou dans la fenêtre de propriétés du rôle.



La caractéristique **Lecture seule** du rôle peut avoir les valeurs suivantes :

- **Ajout seul** : il est toujours possible de relier de nouveaux objets par cette association, mais il n'est pas possible de délier les objets déjà reliés.
- **Lecture seule** : les instances reliées ne peuvent plus être déliées. Il n'est pas possible non plus d'ajouter un nouveau lien.
- **Pas de restriction** : de nouvelles instances peuvent être reliées ou déliées à tout moment sans aucune contrainte.

Ordre d'un rôle

Il est possible de préciser si un rôle est ordonné ou non. Par exemple, pour une commande, il peut être intéressant de mémoriser l'ordre de ses lignes de commande.

Pour spécifier qu'un rôle est ordonné :

1. Ouvrez la fenêtre de **Propriétés** du rôle.
2. Dans l'onglet **Caractéristiques**, cochez la case **Ordonnée**.

Propriété statique d'un rôle

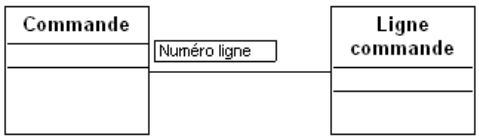
Comme pour un attribut, il est possible de préciser si un rôle peut prendre des valeurs spécifiques pour chacune des instances de la classe ou bien s'il a une valeur qui caractérise l'ensemble de la classe :

1. Ouvrez la fenêtre de propriétés du rôle.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Dans le champ **Statique**, sélectionnez :
   - "Oui" : pour que le rôle ait une valeur qui caractérise l'ensemble de la classe.
   - "Non" : pour que le rôle puisse prendre une valeur différente pour chacune des instances de la classe.

Qualificatif d'un rôle

Un qualificatif est un attribut dont les valeurs partitionnent l'ensemble des objets reliés à un objet à travers une association.

```
Exemple : Une commande comprend plusieurs lignes de
commande. Le numéro de ligne de commande peut servir de
qualificatif pour identifier chacune des lignes.
```

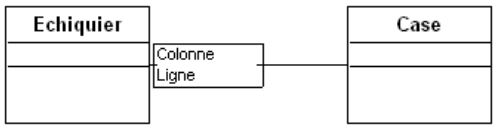


Pour renseigner un qualificatif :

1. Cliquez sur le rôle avec le bouton droit de la souris et sélectionnez **Propriétés**.
   La fenêtre de propriétés du rôle s'ouvre.
2. Cliquez sur l'onglet **Qualificatifs**.
3. Pour ajouter un nouveau qualificatif au rôle, cliquez avec le bouton droit dans la fenêtre.
4. Dans le menu contextuel qui apparaît, activez la commande **Nouveau**.
   Ce même menu contextuel vous permet de modifier les propriétés du qualificatif.

Plusieurs qualificatifs peuvent être nécessaires pour identifier de manière unique chacun des objets de la classe.

Par exemple, chaque case d'un échiquier est identifiée par son numéro de ligne et son numéro de colonne dans l'échiquier.



Surcharger un rôle

Un rôle peut hériter d'un rôle défini au niveau supérieur. La surcharge permet de définir des propriétés supplémentaires sur un rôle hérité.

Pour surcharger un rôle :

1. Ouvrez la fenêtre de propriétés du rôle.
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Caractéristiques**.
3. Dans le cadre **Rôle surchargé**, cliquez sur le bouton **Relier**.
   La fenêtre de sélection apparaît.

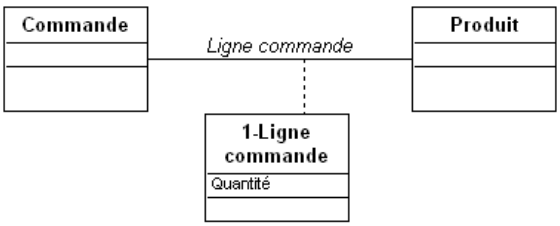
4. Sélectionnez **Propriétés surchargée**. Cette commande affiche les rôles qu'il est possible de surcharger.
5. Sélectionnez le rôle en question et cliquez sur **OK**.

Les classes d'association

Une classe d'association est une association qui possède aussi les propriétés d'une classe comme des attributs.

Il est utile de créer une classe d'association pour préciser des caractéristiques de l'association.

Par exemple, il est nécessaire de préciser la quantité de produit demandée pour chacune des lignes d'une commande.



Pour créer une classe d'association :

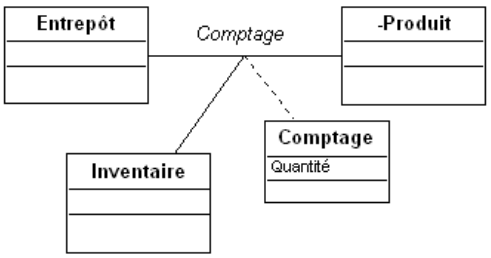
1. Créez une nouvelle classe.
2. A l'aide du bouton **Lien**, créez un lien entre la classe et l'association. La classe d'association est reliée à l'association par un trait pointillé.

> ☛ *Comme pour les classes standard, il est possible de cacher les compartiments et de retailler la classe d'association à l'aide de la commande Affichage de son menu contextuel.*

Définir une association "plus que binaire"

Certaines associations associent non pas deux, mais davantage de classes. Ces associations sont, en principe, rares.

```
Exemple : Lors d'un inventaire, une certaine quantité de
produit a été comptée dans chaque entrepôt.
```

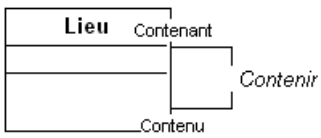


Pour créer une association ternaire :

1. Créez tout d'abord l'association entre deux des classes.
2. Cliquez sur le bouton **Rôle de l'association** 
3. Tirez un lien entre l'association et la troisième classe.

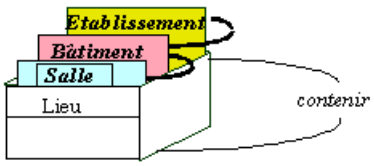
Vous pouvez ensuite créer la classe d'association éventuelle comme précédemment.

Les associations réflexives



Certaines associations mettent en jeu plusieurs fois la même classe.

Une salle de classe, un bâtiment, un établissement scolaire sont tous des lieux.



Une salle de classe est contenue dans un bâtiment, lui-même contenu dans un établissement scolaire.

Une association réflexive porte deux fois sur la même classe.

Créer une association réflexive

Pour créer une association réflexive :

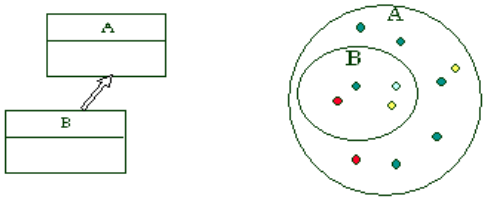
1. Cliquez sur le bouton **Association**  de la barre d'outils.
2. Cliquez dans la classe concernée et faites glisser la souris en dehors de cette classe, puis revenez-y ; relâchez enfin votre pression.
   L'association réflexive apparaît sous forme d'un crochet.

   ☛ *Dans le cas d'une association entre une classe et elle-même, il est indispensable de préciser les rôles afin de distinguer les liens correspondants dans le dessin.*

LES GÉNÉRALISATIONS

Une généralisation représente une relation d'héritage entre une classe générale et une classe plus spécifique. La classe spécifique est cohérente avec la classe plus générale et en hérite ses caractéristiques et son comportement. Elle comporte cependant des informations supplémentaires. Toute instance de la classe spécifique est aussi une instance de la classe générale.

Qu'est-ce qu'une généralisation



La classe A est une généralisation de la classe B. Cela suppose que tous les objets de la classe B sont aussi des objets de la classe A. Autrement dit, B est un sous-ensemble de A.

B est alors la sous-classe, A la super-classe.

Exemple A : Personne, B : Parisien.

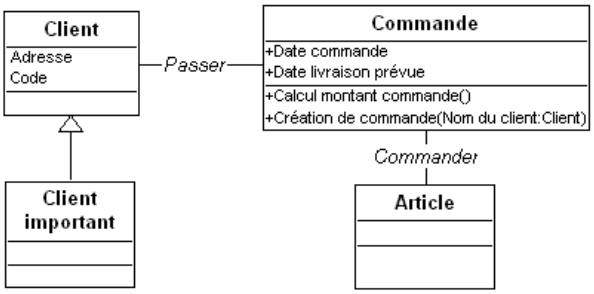
B étant un sous-ensemble de A, les objets de la classe B "héritent" des caractéristiques de ceux de la classe A.

Il n'est donc pas nécessaire de décrire de nouveau pour la classe B :

- Ses attributs
- Ses opérations
- Ses associations

Exemple

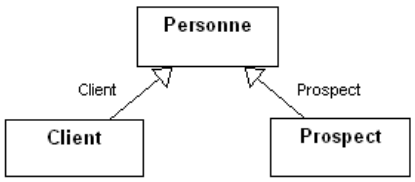
La classe "Client important" qui représente les clients dont le "C.A. sur les 12 derniers mois" dépasse 1 MF, peut être une spécialisation de la classe client (origine).



Dans l'exemple qui précède, les associations et les attributs spécifiés pour "Client" sont aussi valables pour "Client important".

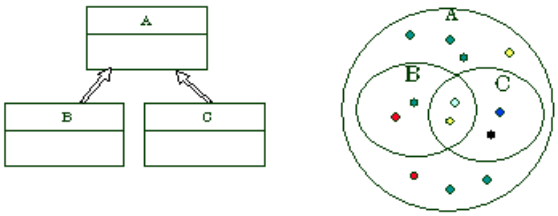
Autres exemples de généralisations :

- Prospect et client sont deux sous-classes de "personne".



- Commande export est une sous-classe de la classe "commande".
- Personne physique et personne morale sont deux sous-classes de la classe "personne".
- Polygone, ellipse et cercle sont des sous-classes de la classe "forme".
- Chêne, orme, et bouleau sont des sous-classes de la classe "arbre".
- Véhicule à moteur, véhicule tout-terrain et véhicule amphibie sont des sous-classes de la classe "véhicule".
- Camion est une sous-classe de la classe "véhicule à moteur".

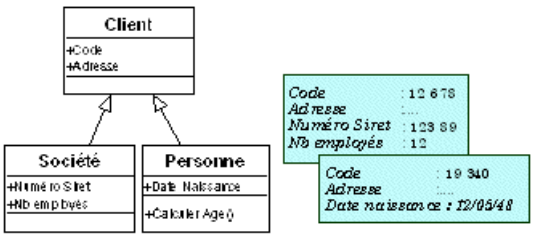
Cas de plusieurs sous-classes



Plusieurs sous-classes d'une même classe :

- Ne sont pas forcement exclusives.
- Ne forment pas nécessairement une partition.

Intérêt des sous-classes

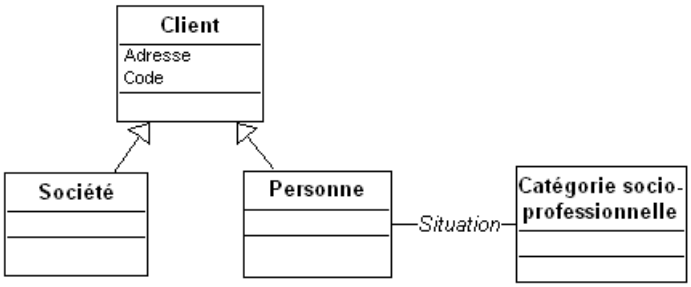


Une sous-classe hérite de tous les attributs, opérations et associations de sa super-classe, mais elle peut avoir des attributs ou des associations que ne possède pas sa super-classe.

Une sous-classe peut ainsi avoir des attributs spécifiques. Ceux-ci n'ont de sens que pour une sous-classe particulière. Dans l'exemple ci-dessus :

- Le "numéro de Siret" et le "nombre d'employés" n'ont de sens que pour une "société".
- La "date de naissance" est caractéristique d'une "personne", pas d'une "société".
- De même, il est utile de calculer l'"âge" d'une "personne". Cet attribut et cette opération n'ont généralement pas d'intérêt pour une "société".

Une sous-classe peut également avoir des *associations* spécifiques.



- Une "personne" entre dans une "catégorie socio-professionnelle" : "cadre", "employé", "commerçant", "agriculteur", etc. Cette classification n'a pas de sens pour une "société". (Il existe également une classification pour les entreprises, mais ce n'est pas la même que pour les personnes.)

Héritage Multiple

Il est parfois utile de spécifier une classe ayant plusieurs super-classes. La sous-classe hérite alors de toutes les caractéristiques des deux super-classes. Cette possibilité doit être utilisée avec précaution.

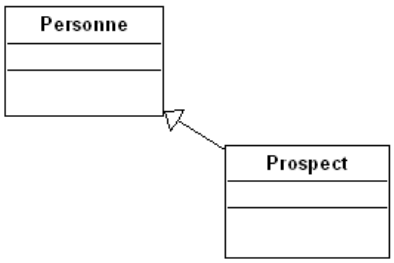
☛ *L'héritage multiple n'est pas pris en compte pour la génération des tables.*

Créer une généralisation

Pour créer une généralisation :

1. Cliquez sur le bouton **Généralisation**  de la barre d'outils
2. Cliquez dans la sous-classe et faites glisser la souris jusqu'à la super-classe, avant de relâcher votre pression.

La généralisation apparaît dans le diagramme, matérialisée par une flèche.

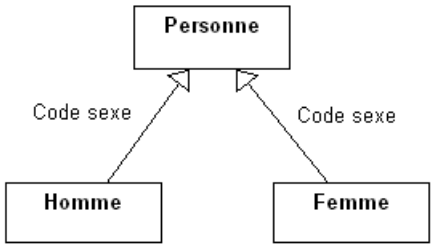


Vous pouvez également créer une généralisation à partir de l'éditeur de classes. Voir "Créer des objets dans l'éditeur de classes", page 85.

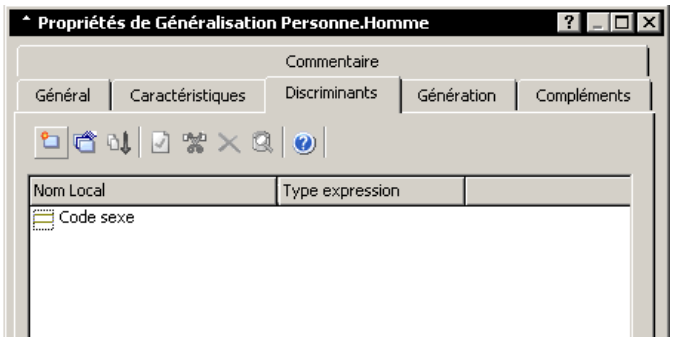
Discriminant

Le discriminant est l'attribut d'une généralisation dont la valeur permet de répartir les objets entre les sous-classes associées à la généralisation.

Par exemple, l'attribut code-sexe permet de répartir les objets de la classe personne entre la sous-classe homme ou femme.



Vous pouvez définir le ou les discriminants dans la fenêtre de propriétés de la généralisation.



☛ *Il est également possible de préciser si une généralisation est :*

- Disjointe : Une instance ne peut pas appartenir simultanément à deux sous-classes de cette généralisation.

- Complète : Toutes les instances de la super-classe appartiennent à au moins une des sous-classes de cette généralisation.

SPÉCIFIER LES INTERFACES

Une interface représente la partie visible d'une classe ou d'un paquetage dans une relation contractuelle de type client - fournisseur. L'interface est un stéréotype de classe.

Une interface est constituée d'un ensemble d'opérations qui décrivent le comportement d'un élément. En particulier, une interface représente la partie visible d'une classe ou d'un paquetage dans une relation contractuelle de type client - fournisseur.

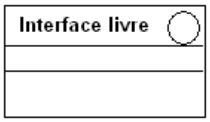
Ce sont des interfaces entre les différents composants du système informatique.

Créer une interface

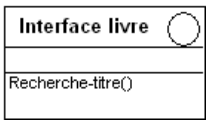
Pour créer une classe interface :

1. Sélectionnez le bouton **Interface** dans la barre d'outils et cliquez dans le dessin.
2. Saisissez son nom dans la fenêtre qui apparaît.

La classe interface apparaît dans le dessin avec un rond distinctif en haut à droite.



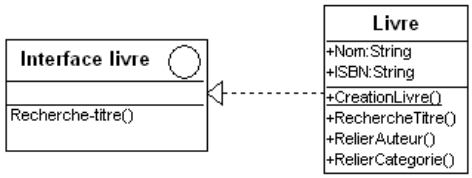
Vous pouvez spécifier les opérations de l'interface comme pour n'importe quelle classe.



Pour préciser qu'une interface est supportée par une classe :

1. Cliquez sur le bouton

2. Effectuez le lien en partant de la classe fournisseur pour aller vers l'interface supportée.



Pour indiquer qu'une classe requiert une interface :

1. Cliquez sur le bouton 
2. Effectuez le lien en partant de la classe cliente vers l'interface requise.

SPÉCIFIER LES DÉPENDANCES

Vous pouvez créer des paquetages à l'aide du bouton  de la barre d'objets.

☛ *Le bouton* ***Vues***  *vous permet de préciser les boutons que vous voulez voir apparaître dans la barre d'objets.*

Pour indiquer qu'un paquetage référence une classe ou un autre paquetage :

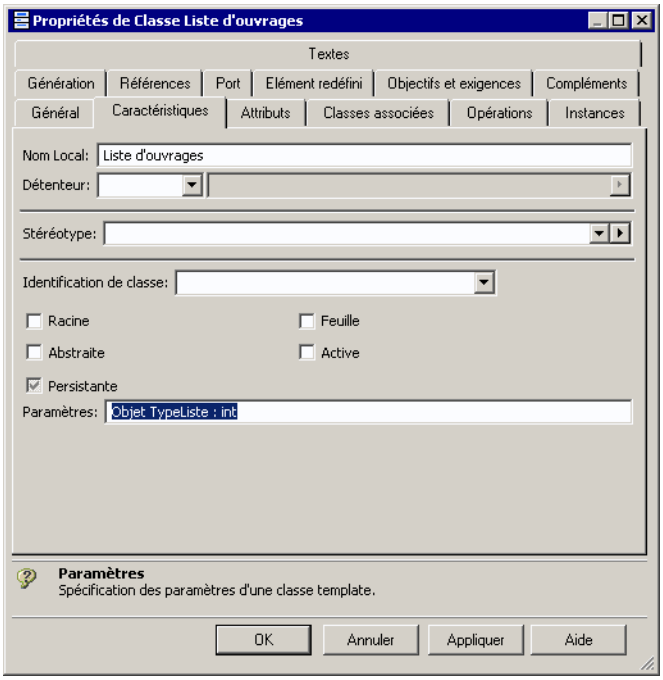
1. Cliquez sur le bouton 
2. Effectuez le lien en partant d'un paquetage vers le paquetage ou la classe qu'il référence.

SPÉCIFIER DES CLASSES PARAMÉTRÉES

Une classe paramétrée permet de définir des caractéristiques et un comportement modulable en fonction de la valeur de certains paramètres. Par exemple, une classe paramétrée peut être utilisée pour gérer des listes d'objets. Le paramètre sera alors le type d'objet que l'on veut gérer sous forme de liste. Ce type de classe est implémenté en particulier dans le langage C++.

Pour spécifier une classe paramétrée **:**

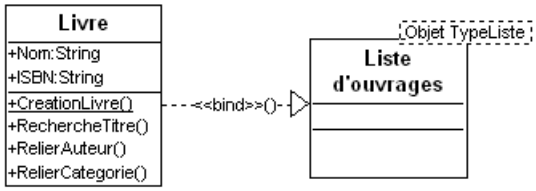
1. Ouvrez la fenêtre des **Propriétés** de la classe et cliquez sur l'onglet **Caractéristiques**.
2. Vous pouvez saisir les paramètres et préciser éventuellement leur type.



Les paramètres de la classe s'affichent en haut à droite.

Pour relier une classe à une classe paramétrée :

- Cliquez sur le bouton 
- Créez le lien en partant de la classe pour aller vers la classe paramétrée.



LES CONTRAINTES

Une contrainte est une déclaration qui établit un contrôle ou une règle de gestion impliquant généralement plusieurs classes.

La plupart des contraintes impliquent les associations entre les classes.

Exemples de contraintes :

- La personne responsable d'un service doit appartenir à ce service.
- Toute commande facturée doit avoir été livrée auparavant.
- La date de livraison doit être postérieure à la date de commande.

Un capteur couvrant une zone ne peut déclencher qu'une alarme protégeant cette même zone.

Pour créer une contrainte :

1. Cliquez sur le bouton **Contrainte**  de la barre d'objets.

   ☛ *S'il n'est pas affiché, cliquez sur le menu **Affichage** > **Vues et détails** et cochez la case "Contraintes".*

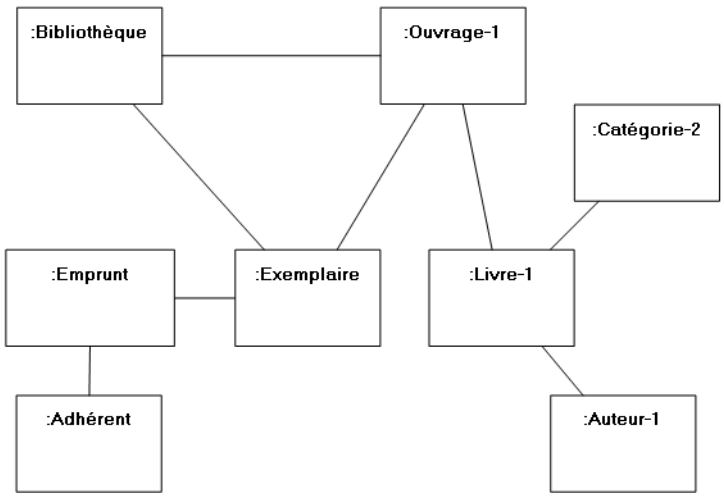
2. Cliquez dans une des associations concernées par la contrainte, et faites glisser la souris jusqu'à la deuxième association, avant de relâcher votre pression.
   La fenêtre d'ajout d'une contrainte s'ouvre.
3. Saisissez le nom de la contrainte puis cliquez sur **Créer**.
   La contrainte apparaît dans le diagramme.

   ☛ *Vous pouvez relier une contrainte à d'autres classes ou associations à l'aide du bouton **Lien***

LE DIAGRAMME D'OBJETS

Un diagramme d'objets, ou diagramme d'instances, contient des objets avec des valeurs exemples pour leurs attributs, et des liens. Il montre en détail l'état du système à un instant précis.

Vous pouvez créer le diagramme d'objets d'une classe, d'un composant, d'un paquetage ou d'un cas d'utilisation.



Les objets

Un objet est une entité avec une identité et des frontières clairement définies dont l'état et le comportement sont encapsulés. Son état est défini par les valeurs de ses attributs et de ses liens avec d'autres objets. Son comportement est représenté par ses opérations et ses méthodes. Un Objet est une instance de Classe.

Exemples d'objets :

- Objets de gestion :
  - Jacques Dupond, Pierre Durand, Paul Smith sont des instances de la classe personne.
  - Les commandes no 10533 et 7322 sont des instances de la classe commande.
  - Ecran Sony SPD-1730, Compaq Deskpro 200 sont des instances de la classe article.
  - Dupond de Nemours, Burger King sont des instances de la classe société.
- Objets techniques utilisés pour la programmation :
  - Dlg_Order_Create, Dlg_Customer_Query sont des instances de la classe fenêtre.
  - Str_Customer_Name, Str_Product_Comment sont des instances de la classe chaîne.

> ☛ *Les objets représentés dans un diagramme d'objets peuvent être des instances de classe, de paquetage, de cas d'utilisation, de composant ou de nœud, ce qui permet de définir des diagrammes de séquence au niveau de détail souhaité.*

Créer un objet (une instance)

Pour créer un objet :

1. Cliquez sur le bouton  **Instance**.
   Vous pouvez créer des objets de différents types. La flèche située à droite du bouton offre un raccourci vers les types d'objets Classe ou Composant, plus fréquemment utilisés.
2. Puis cliquez sur le plan de travail du diagramme.
   La fenêtre permettant d'ajouter d'une instance s'ouvre.
3. Saisissez le **Nom** de l'instance.
4. Précisez si nécessaire le **Type d'instance**.
5. Cliquez sur **Créer**.

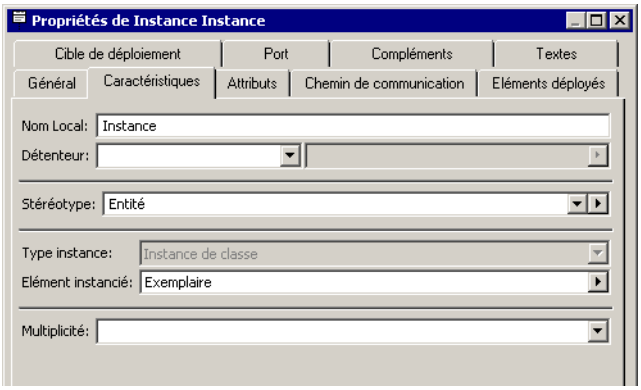
L'instance est posée dans le diagramme.

Propriétés d'une instance

Pour ouvrir la fenêtre de propriétés d'une instance :

- Cliquez avec le bouton droit sur l'instance et sélectionnez **Propriétés**.

Plusieurs onglets permettent de définir les propriétés d'une instance.



Vous pouvez :

- Sélectionner le **Type** de l'instance (Acteur, Classe, etc.).
- Préciser de quelle **Classe**, quel **Acteur**, etc. cet objet est une instance.
- Indiquer un nom pour cette Instance.
- Préciser son **Stéréotype**.

Valeur d'un attribut

Pour renseigner la valeur d'un attribut :

1. Cliquez avec le bouton droit sur l'instance de la classe contenant l'attribut.
2. Sélectionnez **Attributs**.
3. Dans la fenêtre qui apparaît, indiquez la valeur de l'attribut. Vous pouvez renseigner une valeur instanciée ou une valeur constante.
   - **Valeur instanciée** : cliquez dans cette colonne pour afficher la liste des instances possibles pour l'attribut sélectionné. Il s'agit de valeurs variables.
   - **Valeur** : cliquez dans la colonne et entrez la valeur de l'attribut.

Les liens

Un lien entre objets représente une instance d'association entre deux objets.

Exemples de liens entre objets :

- La commande n° 10733 a été passée par Jacques Dupond.
- La commande 10733 comprend les produits Ecran Sony SPD-1730 et Compaq Deskpro 200.
- Mr Jacques Dupond travaille pour la société Dupond de Nemours.
- La fenêtre Dlg_Customer_Query affiche la chaîne de caractères Str_Customer_Name.

Créer un lien

Pour créer un lien :

1. Cliquez sur le bouton **Lien**  de la barre d'outils.
2. Cliquez dans un des objets concernés, et faites glisser la souris jusqu'au deuxième objet, avant de relâcher votre pression.

Le lien apparaît alors dans le dessin.

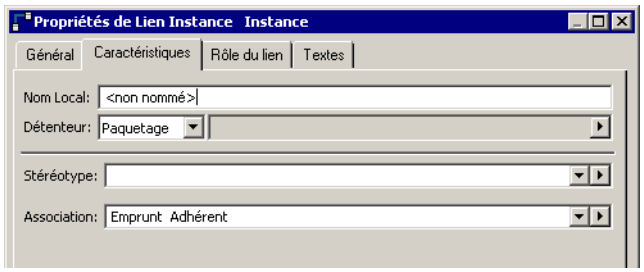
Si un lien entre les deux objets existe déjà, une fenêtre s'ouvre pour vous permettre de choisir parmi les liens existants ou en créer un nouveau.

Propriétés d'un lien

Pour ouvrir la fenêtre de propriétés d'un lien :

- Cliquez avec le bouton droit au centre du lien et sélectionnez **Propriétés**.

La fenêtre de propriétés s'affiche.



> ☛ *Si vous ne cliquez pas au centre du lien, c'est la fenêtre Propriétés d'un des rôles qui va s'ouvrir.*

Sous l'onglet **Caractéristiques**, vous pouvez préciser :

- Le **Nom** du lien.
- Le **Stéréotype** du lien.
- L'**Association** correspondant au lien.
- Le **Paquetage** détenteur du lien.

Et sous l'onglet **Rôle du lien** :

- Pour chaque **Instance** reliée par ce lien, le nom du **Rôle** et la **Multiplicité** de ce rôle.

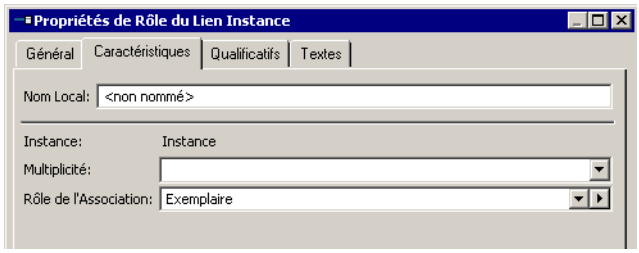
> ☛ *Parmi les associations proposées ne figurent que celles qui figurent entre les classes des deux objets.*

Propriétés d'un rôle

Pour ouvrir la fenêtre de propriétés d'un rôle :

1. Dans la fenêtre de propriétés d'un lien, cliquez sur l'onglet **Rôle du lien**.
2. Cliquez avec le bouton droit sur le rôle et sélectionnez la commande **Propriétés**.

La fenêtre de propriétés du rôle s'affiche.



☛ *Si vous ne cliquez pas sur le rôle, c'est la fenêtre de propriétés du lien qui va s'ouvrir.*

Dans cette fenêtre vous pouvez préciser :

- Un **Nom** pour l'instance de rôle.
- Le **Rôle** de cette instance.
- La **Multiplicité** de l'instance de rôle.
- Pour cette instance de rôle, les valeurs des *qualificatifs* définis au niveau de la classe.

L'ÉDITEUR DE CLASSES

L'éditeur de classes vous permet de visualiser les propriétés d'un paquetage ou d'une classe ainsi que ses liens dans la base.

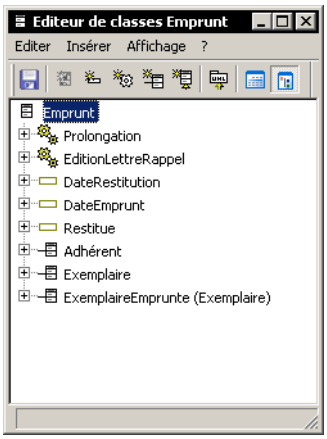
Ouvrir l'éditeur de classes

L'éditeur de classes est accessible à partir d'une classe ou d'un paquetage, et ce depuis le navigateur ou depuis un diagramme.

Pour ouvrir l'éditeur de classes :

1. Cliquez avec le bouton droit sur le paquetage ou la classe.
2. Sélectionnez **Editeur de classes**.

L'éditeur de classes s'affiche. Il présente sous forme d'arborescence le paquetage ou la classe en question ainsi que les objets associés.



Paramètres d'affichage de l'éditeur de classes

Pour personnaliser l'affichage de l'éditeur de classes :

- Cliquez sur le menu **Affichage** > **Barre d'outils...**
- Dans la fenêtre qui apparaît, cochez la case **Barre de filtrage**.

Les boutons apparaissant dans la barre d'outils vous permettent d'afficher :

 les attributs

 les opérations

 les interfaces supportées

 les classes héritées

 les classes héritières

Pour conserver l'éditeur au premier plan :

- Cliquez sur le bouton 

Pour actualiser l'affichage :

- Cliquez sur le bouton 

☛ *Ces boutons peuvent être filtrés. Dans ce cas, cliquez sur le menu* ***affichage*** > ***Barre d'outils...***

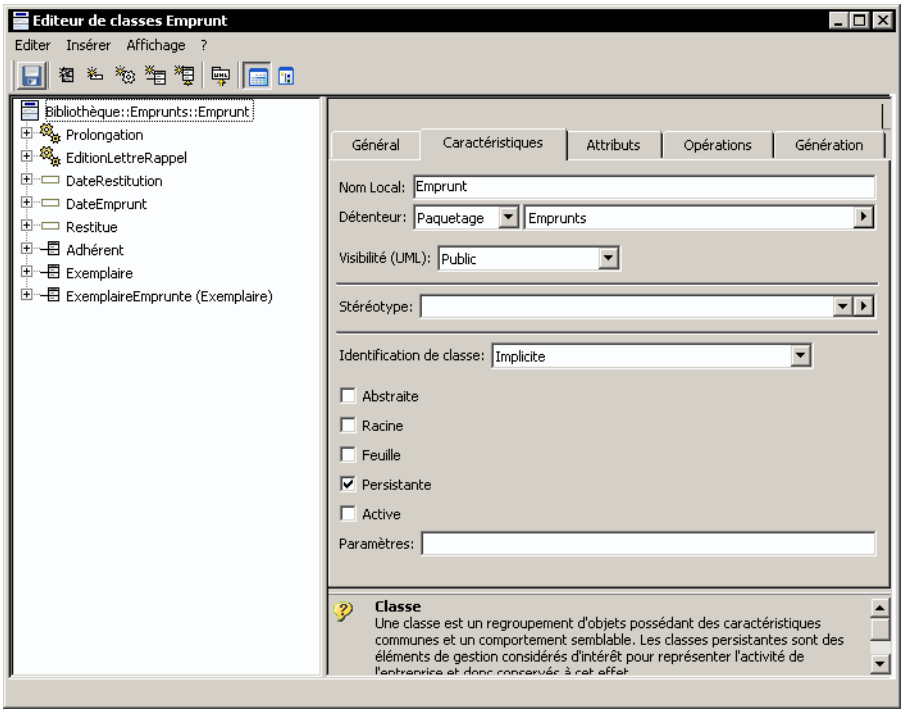
Propriétés des objets

Vous pouvez afficher le détail des objets présentés dans l'arborescence.

Pour voir le détail des objets :

- Cliquez sur le bouton **Détails**  de l'éditeur.

Les propriétés de l'objet sélectionné s'affichent dans la partie droite de la fenêtre.



Tout comme dans la fenêtre de propriétés, différents onglets vous permettent de visualiser et de modifier les propriétés de l'objet.

☛ *Voir "Propriétés d'une classe", page 33.*

Créer des objets dans l'éditeur de classes

L'éditeur de classes vous permet de créer des attributs, des opérations, des associations et des héritages.

Par exemple, pour créer une opération :

1. Cliquez dans l'arborescence sur la classe concernée.
2. Cliquez sur le bouton **Création d'une opération**  dans la barre d'outils de l'éditeur.
   La fenêtre **Création d'une opération** apparaît.
3. Entrez le nom de l'opération puis cliquez sur **OK**.

L'opération créée apparaît à la fois dans l'arborescence et sous l'onglet **Opération** de la fenêtre de propriétés de la classe.

Créer une association entre deux classes

Pour créer une association entre deux classes :

1. Sélectionnez dans l'éditeur la classe pour laquelle vous voulez créer une association.

   ☛ *Vous pouvez également créer une association à partir d'un rôle.*

2. Cliquez sur le bouton **Création d'une association**  de la barre d'outils.
3. Dans la fenêtre qui apparaît, saisissez le nom de la classe associée. Vous pouvez rechercher une classe existante ou en créer une nouvelle.
4. Cliquez sur **OK**.

L'association apparaît dans l'éditeur.

Créer une généralisation entre deux classes

Pour créer une généralisation entre deux classes :

1. Sélectionnez dans l'éditeur la classe pour laquelle vous voulez créer une généralisation.
2. Cliquez sur le bouton **Création d'un héritage**  de la barre d'outils.
3. Dans la fenêtre qui apparaît, saisissez le nom de la classe dont vous voulez hériter. Vous pouvez rechercher cette classe en cliquant sur la flèche noire située à l'extrémité du champ.
4. Cliquez sur **OK**.

La sur-classe apparaît dans l'éditeur.

GÉNÉRER UN DIAGRAMME DE CLASSES

Vous pouvez générer un diagramme de classes depuis :

- Un paquetage
- Une classe

Depuis un paquetage

1. Cliquez avec le bouton droit sur le paquetage.
2. Sélectionnez **Nouveau > Générer un diagramme de classes**.

Cette commande construit un diagramme descriptif du paquetage. Il contient toutes les classes détenues par le paquetage, et pour chaque classe :

- Un niveau d'héritage.
- Le premier niveau d'association avec les éventuelles classes associatives.
- Le premier niveau d'interfaces implémentées.

Depuis une classe

Vous pouvez générer deux types de diagrammes de classes :

- **Diagramme de classes simple** : il décrit pour la classe en question le premier niveau de classes associées avec les éventuelles classes associatives, le premier niveau d'héritage et le premier niveau d'interfaces implémentées.
- **Diagramme de classes élaboré** : il décrit pour la classe en question ***n*** niveaux de classes associées avec les éventuelles classes associatives (la récursion s'arrête si une classe C n'appartient pas au même paquetage que la classe décrite mais si C est incluse dans le diagramme), le premier niveau d'héritage et le premier niveau d'interfaces implémentées.

Fonctionnalité de réorganisation automatique

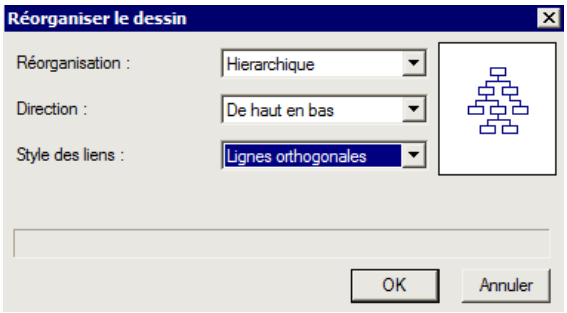
Le graphe des utilisateurs, le diagramme de classes, le diagramme de collaboration, le diagramme de composants et le diagramme relationnel comportent la fonctionnalité de réorganisation du dessin.

> ☺ *La fonctionnalité de réorganisation est déclenchée automatiquement au chargement d'un diagramme qui ne comporte pas encore de dessin.*

Pour modifier l'organisation d'un dessin existant :

1. Sélectionnez le menu **Dessin > Réorganiser le dessin**.

2. Sélectionnez le mode de réorganisation souhaité, la direction ainsi que le style des liens dans le diagramme.



☺ *L'image miniature à côté des options de réorganisation vous permet d'avoir un aperçu de chaque type de réorganisation.*

3. Cliquez sur **OK** pour appliquer les modifications.

LES DIAGRAMMES DE STRUCTURE ET DE DÉPLOIEMENT

Outre les diagrammes de classes et d'objets, les diagrammes structurels comprennent :

- ✓ Le diagramme de paquetages, qui permet d'organiser les éléments du modèle.
- ✓ Le diagramme de composants, qui met en évidence les relations de dépendance entre composants.
- ✓ Le diagramme de structure composite, qui décrit les interactions entre les composants et leurs parties.

LE DIAGRAMME DE PAQUETAGES

Un diagramme de paquetages permet d'organiser les éléments de modélisation, de manière à assurer une partition des travaux de spécifications et de développement.

Un élément ne doit apparaître que dans un seul paquetage.

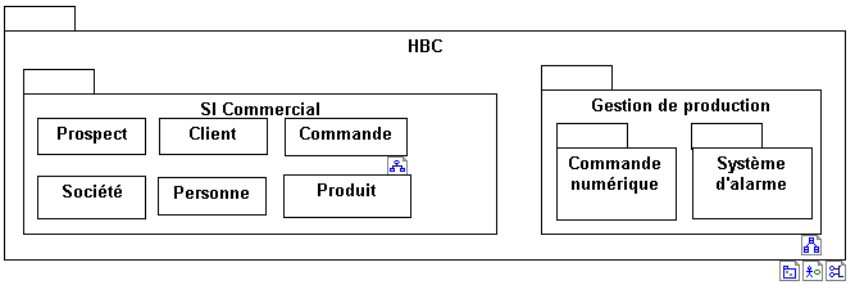
Le découpage en paquetage est généralement fait de manière à minimiser les interactions entre les différents paquetages.

Exemple de diagramme de paquetages

Le paquetage "HBC" contient les paquetages "SI Commercial" et "Gestion de production".

Le paquetage "Gestion de production" se décompose en deux paquetages "Commande numérique" et "Système d'alarme".

Le paquetage "SI Commercial" contient les classes "Prospect", "Client", "Société", "Personne", "Commande" et "Produit".



Créer un diagramme de paquetages

Un diagramme de paquetage se crée depuis un paquetage. Pour la création d'un paquetage, voir "Créer un diagramme de cas d'utilisation", page 14.

Pour créer un diagramme de paquetages :

1. Dans la fenêtre de navigation **Objets principaux**, cliquez avec le bouton droit sur le nom du paquetage.
2. Dans le menu contextuel qui s'affiche, cliquez sur **Nouveau** > **Diagramme**.
   Une fenêtre contenant les différents types de diagramme possibles apparaît.
3. Sélectionnez "Diagramme de paquetages" et cliquez sur **Créer**.

Définir les paquetages

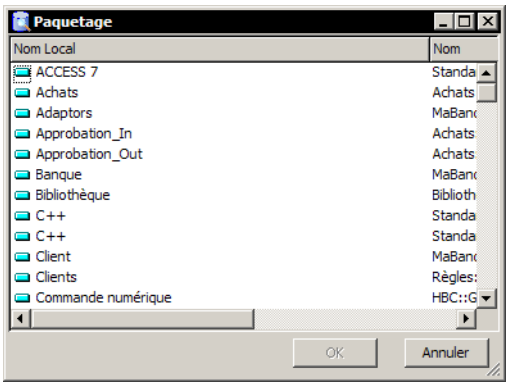
Un paquetage partitionne le domaine d'étude et les travaux associés. Il permet de regrouper divers éléments, en particulier des cas d'utilisations et des classes. Un paquetage peut aussi contenir d'autres paquetages. Les paquetages sont liés entre eux à travers des rapports contractuels définissant leur interface.

Exemples de *paquetages* :

- Le système d'information commercial.
- La comptabilité.
- La gestion de production.
- La commande numérique d'une machine.
- La gestion des stocks.
- La gestion du système d'alarme et du téléphone.

Pour ajouter dans le diagramme un paquetage existant :

1. Dans le diagramme de paquetages, cliquez sur le bouton **Paquetage** de la barre d'objets puis cliquez sur le plan de travail.
2. Dans la fenêtre **Ajout d'un paquetage**, sélectionnez la commande **Lister** à l'aide de la flèche
   La liste des paquetages apparaît.



☺ *Notez que vous pouvez faire apparaître la liste des paquetages existants en utilisant la touche <Ctrl-L>.*

Vous pouvez également saisir le début du nom, par exemple "Paqu", et en utilisant la touche <Ctrl-L>, faire apparaître la liste des paquetages dont le nom commence par "Paqu".

3. Sélectionnez le paquetage qui vous intéresse et cliquez sur **OK**.
   Le nom du diagramme apparaît dans la fenêtre **Ajout d'un paquetage**.
4. Cliquez sur **Relier**.

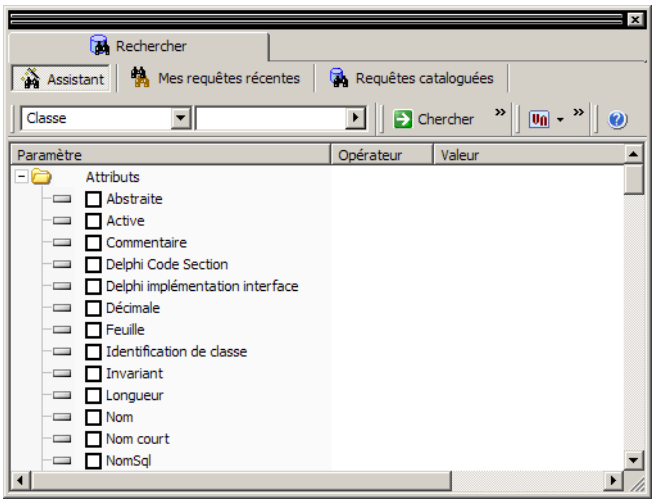
Le paquetage apparaît dans le diagramme.

Définir les classes

Le diagramme de paquetages permet de répartir les classes entre les paquetages.

Pour ajouter les classes dans le diagramme de paquetages :

1. Recherchez les classes déjà créées en cliquant sur le bouton de recherche de la barre **MEGA**.
2. Dans la fenêtre qui s'ouvre, choisissez la cible "Classe".



3. Cliquez sur **Chercher**.
   La liste des classes s'ouvre.
4. Sélectionnez les classes qui vous intéressent et glissez-les dans le diagramme.

Spécifier les dépendances dans un diagramme de paquetage

Des liens vous permettent d'indiquer si un paquetage détient ou référence une classe ou un autre paquetage.

Pour indiquer qu'un paquetage référence une classe ou un autre paquetage :

1. Cliquez sur le bouton
2. Puis, effectuez le lien en partant d'un paquetage vers le paquetage ou la classe qu'il référence.
   Une fenêtre vous demande le type de lien à créer.
3. Sélectionnez "Paquetage référencé" ou "Classe référencée" selon qu'il s'agisse d'un paquetage ou d'une classe.

LE DIAGRAMME DE COMPOSANTS

Un diagramme de *composants* présente l'interdépendance des composants logiciels et des *interfaces* (il définit qui utilise quoi).

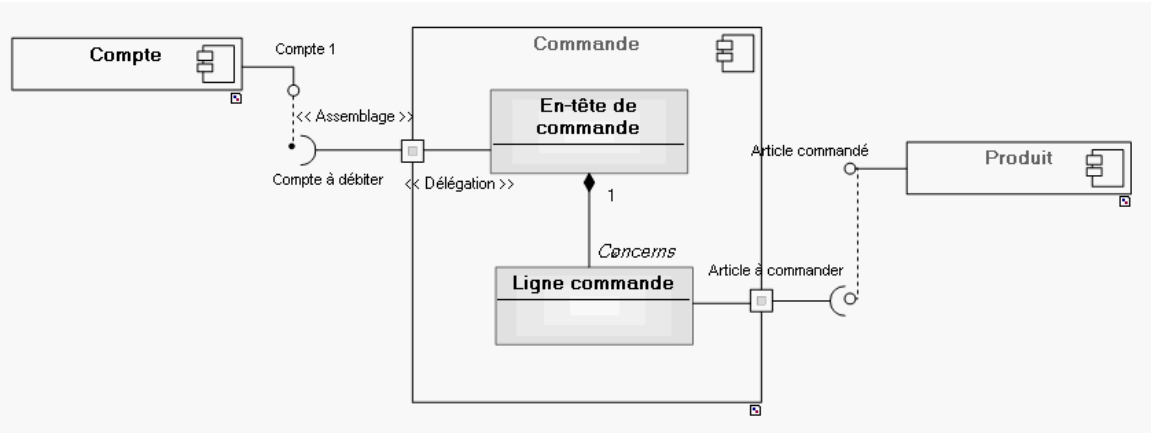
> 🕮 *Un composant représente une partie modulaire d'un système qui encapsule son contenu et qui est remplaçable dans son environnement. Un composant définit son comportement par les interfaces qu'il fournit et celles qu'il requiert.*
>
> 🕮 *Une interface représente la partie visible d'une classe ou d'un paquetage dans une relation contractuelle de type client - fournisseur. L'interface est un stéréotype de classe.*

Un diagramme de composants contient des composants et des classes de stéréotype interface. Il est également possible d'y préciser les paquetages implémentés par les composants.

Vous pouvez créer un diagramme de composants depuis un composant ou un paquetage.

Exemple de diagramme de composants

Ce diagramme décrit les éléments détenus par le composant "Commande" et les interactions de ces éléments avec des composants externes.



Les composants

Un composant représente une partie modulaire d'un système qui encapsule son contenu et qui est remplaçable dans son environnement. Un composant définit son comportement par les interfaces qu'il fournit et celles qu'il requiert.

Un composant peut être remplacé par un autre si leurs interfaces sont conformes.

Un composant peut être un logiciel, un programme, un élément de code, etc.

Il est représenté par l'icône suivante : 

Les interfaces

Créer les interfaces des composants

Une interface représente la partie visible d'une classe ou d'un paquetage dans une relation contractuelle de type client - fournisseur.

L'interface est un type particulier de classe.

Pour créer une classe de stéréotype "Interface" dans le diagramme de structure composite :

1. Cliquez sur le bouton **Interface**  puis cliquez dans le diagramme.
2. Dans la fenêtre qui apparaît, saisissez le nom de la classe.
3. Cliquez sur **Créer**.

> ☛ *Vous pouvez spécifier le détail de l'interface en termes d'attributs et d'opérations dans le diagramme de classes de la même manière que pour une classe.*

Relier les interfaces aux autres objets

Deux types de lien permettent de différencier les interfaces requises des interfaces fournies.

Une interface requise est une interface nécessaire au fonctionnement de l'objet.

```
Exemple : le composant « Gestion des achats » a besoin pour
son fonctionnement de l'interface « Produit » pour pouvoir
associer une commande d'achat aux produits commandés.
```

Une interface fournie est une interface mise à disposition par un objet à destination d'autres objets.

```
Exemple : le composant « Gestion des produits » met à
disposition l'interface « Produit ».
```

Vous pouvez définir les interfaces requises et les interfaces fournies par un objet indépendamment des autres objets.

Pour préciser qu'une interface est supportée par un objet :

1. Cliquez sur le bouton 
2. Dessinez le lien en partant de l'objet fournisseur (un composant, un paquetage ou une classe) pour aller vers l'interface supportée.

Pour indiquer qu'un objet requiert une interface :

1. Cliquez sur le bouton 

2. Dessinez le lien en partant de l'objet client vers l'interface requise.

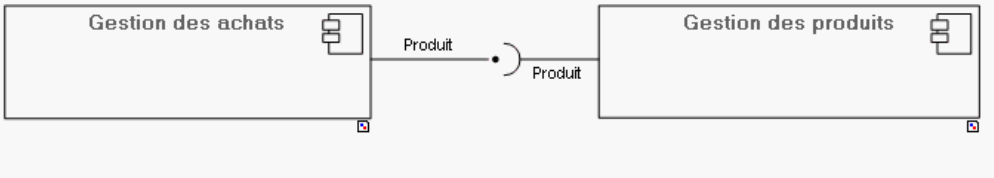
> ☛ *Vous pouvez également spécifier les dépendances entre paquetages ou entre paquetages et classes à l'aide du bouton*  *comme dans le diagramme de paquetages.*

Selon le type de lien, la forme de l'interface change : l'interface requise est représentée par un demi-cercle, l'interface fournie est représentée par un cercle.

Relier des interfaces

Deux interfaces peuvent être reliées l'une à l'autre. Cette connexion est modélisée par un connecteur.

Vous pouvez également indiquer qu'une interface fournie par un objet est requise par un autre. Il s'agit ici d'une seule et même interface.



Les ports

Les ports permettent de connecter un composant à ses parties ou à son environnement.

Les ports sont symbolisés par un carré dans le diagramme, et posés en bordure du composant décrit lorsqu'ils assurent la connexion avec l'extérieur.

Ils sont reliés aux composants par des connecteurs.

Les ports peuvent spécifier les requêtes envoyées et les services fournis par le composant ainsi que les requêtes et services qu'ils peuvent requérir d'autres parties du système. Ces requêtes et services sont représentés par des classes de type Interface.

Vous pouvez visualiser les interfaces associées à un port dans la fenêtre de propriétés d'un port, sous l'onglet **Interfaces fournies et requises**.

Les connecteurs

Les connecteurs permettent de relier les objets du diagramme.

Les connecteurs de type simple ne spécifient aucun type de connexion particulier, ils sont utilisés notamment pour relier les instances d'objets décrits dans des collaborations.

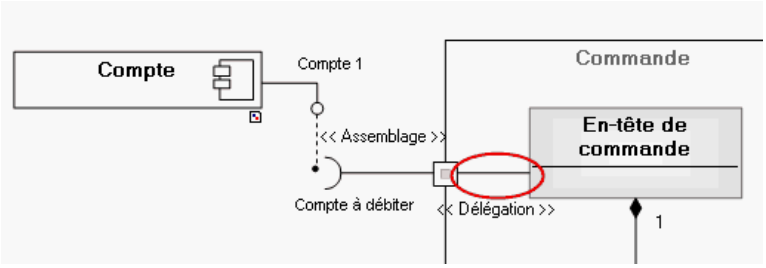
Dans le diagramme de structure composite, il est possible de spécifier le type de connecteur qui relie deux composants : Assemblage ou Délégation.

Connecteur de délégation

Un connecteur de type "Délégation" montre le réacheminement de requêtes vers un élément du composant chargé de les réaliser.

Le lien de délégation peut se faire directement entre le port du composant et l'élément du composant ou entre le port du composant et le port de l'élément.

Ci-dessous, le composant "Commande" délègue la gestion des comptes à débiter à la classe "En-tête de commande".



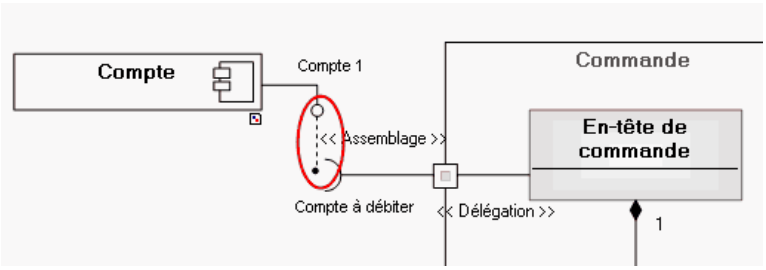
Connecteur d'assemblage

Un connecteur de type "Assemblage" est un connecteur entre deux ou plusieurs composants ou ports qui indique qu'un ou que plusieurs composants fournissent les services que d'autres utilisent.

☛ *Il peut s'agir d'autres objets que de composants.*

Pour relier des ports ou des composants qui partagent une interface, vous pouvez également utiliser les liens "Interface fournie" et "Interface requise".

Un connecteur de type "Assemblage" relie l'interface fournie par le composant "Compte" à l'interface requise par la classe "En-tête Commande".



LE DIAGRAMME DE STRUCTURE COMPOSITE

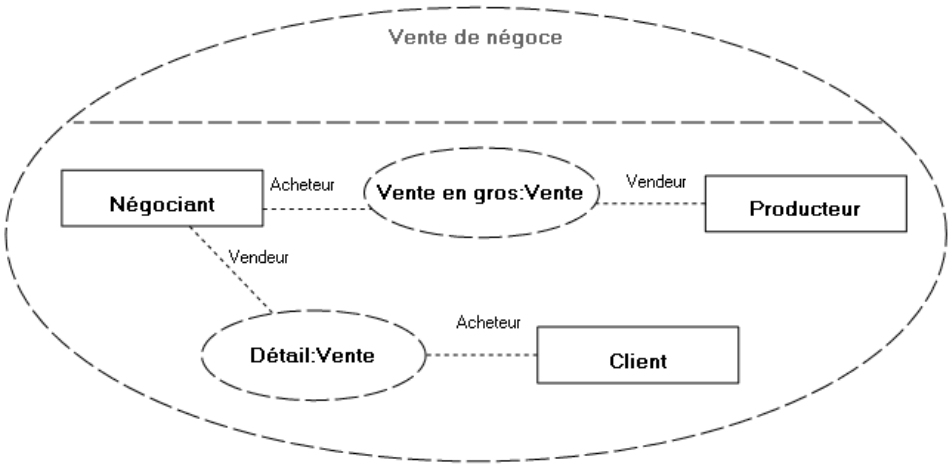
Le diagramme de structure composite permet de décrire la structure interne d'un composant, d'un paquetage ou d'une classe structurée.

Il permet également de préciser les collaborations qui interviennent entre les éléments de la structure dans l'exécution d'une tâche, en mettant en évidence le rôle joué par chaque élément dans la collaboration.

Les éléments de ce diagramme sont les parties (les *parts* en anglais), les ports par le biais desquels les parties interagissent avec l'extérieur, et les connecteurs reliant les parties entre elles ou avec les ports.

Exemple de diagramme de structure composite

Ce diagramme décrit le rôle joué par les parties dans la collaboration "Vente de négoce".



Les parties

Une partie représente un rôle joué par une instance d'une classe ou d'un composant lors de l'exécution d'une tâche.

Les parties sont reliées entre elles par des connecteurs ou des dépendances.

Une partie peut également être reliée - via un connecteur- à un port qui assure l'interface entre le composant décrit et l'extérieur.

Pour plus de détails sur ces éléments, voir :

- ✓ "Les connecteurs", page 97
- ✓ "Les liens de dépendance", page 101
- ✓ "Les ports", page 97.

Les multiplicités définies sur les parties indiquent le nombre d'instances qui sont créées. Les multiplicités sur les rôles de connecteur indiquent le nombre de liens qui peuvent être créés pour chacune de ces instances.

Pour définir la multiplicité d'une partie :

1. Ouvrez la fenêtre de propriétés de la partie.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Cliquez sur la flèche située à l'extrémité du champ **Multiplicité** et sélectionnez la multiplicité voulue.
4. Cliquez sur **OK**.

Les collaborations

Dans le diagramme de structure composite, une *collaboration* décrit les rôles joués par chaque partie (instance) dans la réalisation d'une tâche.

> 🕮 *Une collaboration (UML) décrit une structure collaborative entre plusieurs éléments (rôles), qui accomplissent chacun une fonction spécialisée et qui réalisent collectivement une fonctionnalité attendue du système. Son objectif est de montrer comment un système fonctionne indépendamment d'une utilisation spécifique. On en retirera donc généralement l'identité précise des classes ou des instances qui y participent.*

Elle est représentée par un ovale en pointillé contenant les instances de la collaboration.

Ces instances sont reliées entre elles par des *connecteurs*. A chaque extrémité du connecteur s'affiche le rôle qui correspond au nom de l'instance.

> 🕮 *Un connecteur est un lien qui permet d'établir une communication entre plusieurs objets. Un connecteur de délégation relie le contrat externe de l'objet (tel qu'il est spécifié par ses ports et/ou ses interfaces) aux objets internes qui vont le réaliser. Un connecteur d'assemblage entre plusieurs objets (ou leurs ports) permet de spécifier comment un des objets fournit l'interface requise par un autre.*

Il est possible d'appliquer le modèle d'une collaboration à différentes instances.

Utilisation de collaboration

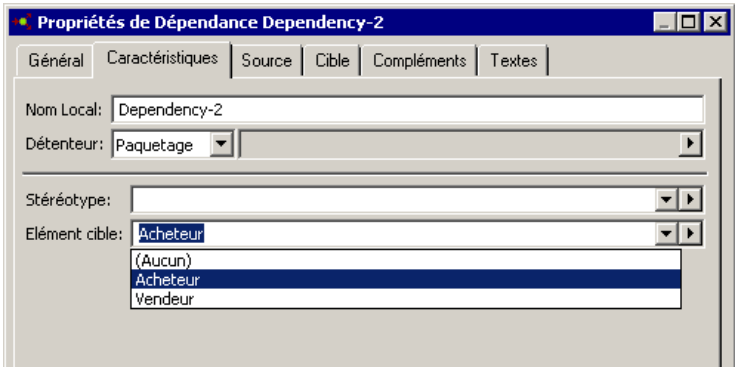
Une utilisation de collaboration représente l'application de la structure décrite par une collaboration à une situation particulière mettant en oeuvre des classes ou des instances spécifiques. Ces classes ou ces instances jouent alors les rôles définis dans la collaboration.

Les instances sont reliées à l'utilisation d'une collaboration par un lien de *dépendance* sur lequel doit être précisé le rôle joué par l'instance.

> 🕮 *Une dépendance précise que l'implémentation ou le fonctionnement d'un ou de plusieurs éléments nécessite la présence d'un ou de plusieurs autres éléments. Il existe plusieurs stéréotypes de dépendance.*

Exemple d'utilisation de collaboration

Dans le cas d'une demande d'achat entre deux instances d'acteur, une collaboration est utilisée. Cette collaboration relie deux rôles : le rôle d'acheteur et le rôle de vendeur. Sur la dépendance qui relie chaque instance à la collaboration, vous pouvez indiquer le rôle que joue l'instance.



Les liens de dépendance

Une dépendance précise que l'implémentation ou le fonctionnement d'un ou de plusieurs éléments nécessite la présence d'un ou de plusieurs autres éléments.

Une dépendance est également une relation de type fournisseur / client qui indique quel est l'élément source et l'élément cible dans la collaboration.

Un stéréotype sur la dépendance permet de spécifier la nature de la dépendance :

- Binding : le binding est une relation entre un template et un élément de modélisation généré à partir du template. Il inclut une liste d'arguments en correspondance avec les paramètres du template.
- Derive : indique une relation de dérivation entre des éléments de modélisation qui sont généralement, mais pas nécessairement, de même type. Une telle relation de dépendance implique que l'un des éléments peut être calculé à partir de l'autre.
- Mapping UML/XML : une expression de mapping qui définit la relation entre les éléments (classes, attributs, ...) d'un schéma ou d'un

diagramme de classes et ceux d'un autre schéma ou diagramme de classes.

- Refine : spécifie une relation de dépendance entre des éléments de modélisation à différents niveaux sémantiques, tels que l'analyse et la conception.
- Trace : spécifie une relation de traçabilité entre des éléments de modélisation ou des ensembles d'éléments de modélisation qui représentent le même concept dans différents modèles.

Pour préciser la nature d’une dépendance :

1. Ouvrez la fenêtre de propriétés de la dépendance.
2. Cliquez sur l’onglet **Caractéristiques**.
3. Dans le champ **Stéréotype**, déroulez la liste et sélectionnez l’un des stéréotypes proposés.

   La flèche vous permet également de créer de nouveaux stéréotypes.

LE DIAGRAMME DE MACHINE À ÉTATS

Un diagramme de machine à état permet de décrire les comportements possibles d'un objet, suivant les événements auxquels il est soumis au cours de son cycle de vie.

Les points suivants sont abordés ici :

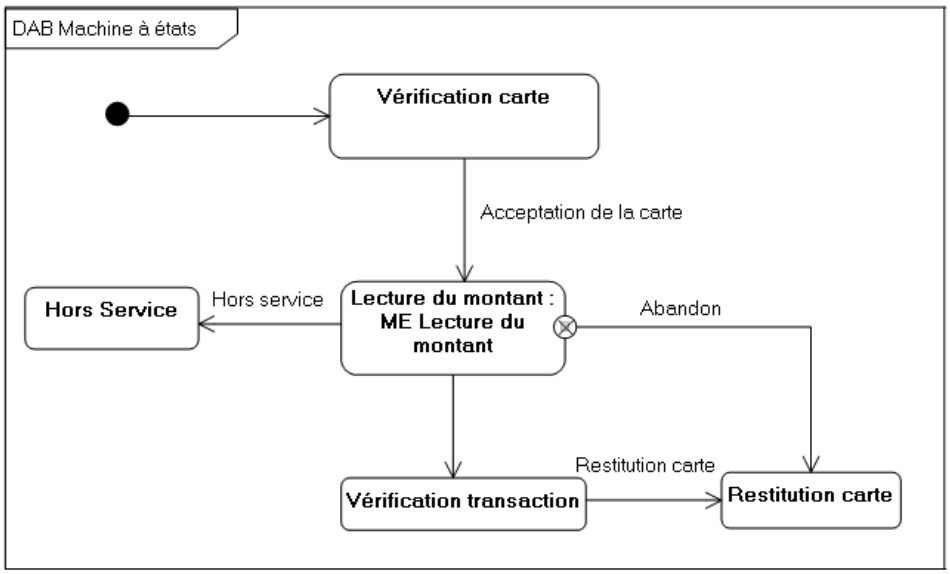
PRÉSENTATION DU DIAGRAMME DE MACHINE À ÉTATS

Une machine à état est l'ensemble des états et des transitions entre états qui définissent le cycle de vie d'un objet variable dans le temps.

Le diagramme de machine à état permet de représenter cet enchaînement d'états que peut prendre un objet en réponse aux interactions avec les objets (internes ou externes au système étudié) qui peuplent son environnement.

Exemple de diagramme de machine à d'états

Le diagramme suivant décrit les comportements possibles d'un distributeur automatique.



Créer un diagramme de machine à états

Un diagramme de machine à états se crée depuis une machine à état.

Vous pouvez créer une machine à état depuis un paquetage, une classe ou un composant.

Pour créer un diagramme de machine à états :

1. Cliquez avec le bouton droit sur une machine à état.
2. Sélectionnez **Nouveau > Diagramme d'états**.

Le diagramme est initialisé par la création d'une région. Une région est une partie d'un état composite ou d'une machine à états qui contient des états et des transitions et dont l'exécution est autonome.

LES ÉTATS

Un état d'objet est une condition ou une situation au cours de la vie d'un objet durant laquelle il satisfait à certaines conditions, exerce une certaine activité ou attend un événement. Un état d'objet représente un intervalle de temps dont les bornes sont deux événements. Un état d'objet est une phase par laquelle passe l'objet au cours de son cycle de vie.

Exemples d'état d'objet

- Une personne peut être :
  - Célibataire
  - Mariée
  - Divorcée
- Un article peut être :
  - Disponible
  - En stock
  - En alerte
  - En rupture de stock.
  - Etc.

Créer un état

Pour créer un état :

1. Cliquez sur la flèche noire associée au bouton **Etat**  de la barre d'insertion du diagramme.
2. Sélectionnez un type d'état.
3. Cliquez sur le plan de travail.
   La fenêtre **Ajout d'un état** s'ouvre.
4. Indiquez le **Nom** de l'état et cliquez sur **Créer**.
   L'état apparaît dans le diagramme.

Les types d'état

Il est nécessaire de préciser le type de l'état lors de sa création. Ce peut être :

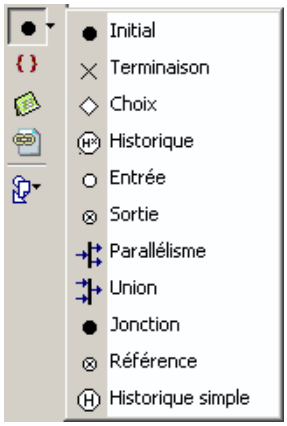
-  Un état normal : ne possède pas de sous-structure.
-  Un état composite : se compose de plusieurs états, décrits dans le diagramme.
-  Une sous-machine à état : appelle la description d'une machine à état décrite par ailleurs. Voir "Précision comportementale d'un état", page 110.
-  Un état final

Lorsque vous posez un état dans un autre, il est automatiquement relié comme composant de cet état.

Pseudo-états

Les pseudo-états sont utilisés pour spécifier des chemins complexes en combinant plusieurs transitions entre états.

Ils peuvent être de différents types : initial, terminaison, choix, historique (deep history), historique simple (shallow history), point d'entrée, point de sortie, parallélisme (fork), union (join), jonction ou référence.



Initial

Le pseudo-état initial a une seule transition en sortie vers l'état Initial de l'objet lors de sa création.

Historique

Un pseudo-état Historique représente la dernière configuration active de l'état composite qui le contient ; c'est-à-dire, la configuration active quand l'état composite a été quitté pour la dernière fois.

Historique simple

Un pseudo-état historique simple représente le plus récent sous-état actif d'un état composite (sans les sous-états de ce sous-état).

Parallélisme

Un parallélisme (fork) sépare une transition en plusieurs transitions concurrentes.

Union

Une union (join) est le regroupement de plusieurs transitions en une seule.

Choix

Représente le choix d'une transition entre plusieurs transitions possibles.

Jonction

Une jonction est utilisée pour définir des chemins de transition complexes entre plusieurs états.

Entrée

C'est un point d'entrée d'une machine à état ou d'un état composite.

Sortie

C'est un point de sortie d'une machine à état ou d'un état composite.

Référence

C'est une référence à une entrée ou à une sortie d'une machine à état ou d'un état composite

Terminaison

L'entrée dans ce pseudo-état implique une terminaison complète de la machine à état.

Historique

Un état **Historique** représente la dernière configuration active d'un état composite ; c'est-à-dire la configuration active quand l'état composite a été quitté pour la dernière fois.

Un état **Historique simple** représente le plus récent sous-état actif de l'état composite.

Exemple :

Prenons l'état "Marié" comme dernière configuration active. Cet état a pour sous-états "Avec enfants" et "Sans enfants". Dans le cas d'un historique, le sous-état "Avec enfants" ou "Sans enfant" est précisé. Dans le cas d'un historique simple, seul l'état "Marié" est pris en compte.

Précision comportementale d'un état

Un état peut être composé de sous-états.

Pour décrire la composition d'un état dans un diagramme :

1. Ouvrez le menu contextuel d'un état et cliquez sur la commande **Nouveau** > **Précision comportementale**.
   La fenêtre de création d'un diagramme de machine à états apparaît.
2. Cliquez sur **Créer**.
   Le diagramme correspondant s'ouvre.

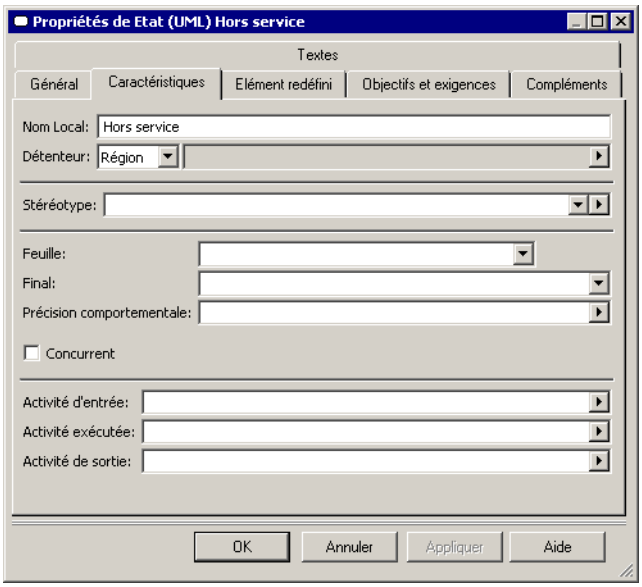
Vous pouvez également définir la composition d'un état en lui associant une machine à état, nouvelle ou existante :

1. Ouvrez la fenêtre de propriétés de l'état décrit.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Dans le champ **Précision comportementale**, créez une machine à état ou recherchez une machine à état existante.

Propriétés d'un état

Pour accéder aux propriétés d'un état :

1. Cliquez avec le bouton droit sur l'état.
2. Sélectionnez **Propriétés**.
   La fenêtre de propriétés de l'état s'ouvre.



Elle vous permet :

- De modifier le **Nom** de l'état.
- D'indiquer si ses sous-états sont **Concurrents**, c'est-à-dire s'ils peuvent être exécutés simultanément ou non.
- D'indiquer la **Précision comportementale** (dans le cas d'un état complexe). Voir "Précision comportementale d'un état", page 110.
- De préciser les **Activités** qui peuvent être effectuées en entrée, en sortie ou pendant que l'objet est dans cet état.

> ☛ *Le contenu de la fenêtre de propriétés d'un état varie en fonction du type de l'état.*

LES TRANSITIONS ENTRE ÉTATS

Le passage d'un état à un autre est matérialisé par une *transition*.

> *Une transition est le passage d'un objet d'un état dans un autre. Une transition est une réponse d'un objet à un événement qu'il reçoit. Quand un événement se produit et que certaines conditions sont satisfaites, l'objet va effectuer certaines actions tandis qu'il est encore dans le premier état puis passer au deuxième état.*

Vous devez définir toutes les transitions qui sont autorisées. Celles qui ne sont pas définies sont interdites.

Exemples de transitions :

En ce qui concerne l'état civil d'une personne, certaines transitions sont possibles :

- Elle peut passer de l'état "célibataire" à l'état "marié".
- Elle peut passer de l'état "marié" à l'état "divorcé".

D'autres ne sont pas possibles :

- Elle ne peut pas passer de l'état "célibataire" à l'état "divorcé".

Créer une transition

Pour créer une transition entre deux états :

1. Cliquez sur le bouton **Transition (UML)**  de la barre d'insertion.
2. Cliquez sur l'état de départ et déplacez la souris jusqu'à l'état d'arrivée.
3. Relâchez le bouton : la transition est créée.

Les types de transition

Une transition peut être de type externe, interne ou locale.

Vous pouvez préciser le type de la transition dans la fenêtre de propriétés de la transition, sous l'onglet **Caractéristiques**.

Transition externe

Une transition externe est une transition qui modifie l'état actif.

Transition interne

Une transition interne à un objet permet de prendre en compte l'arrivée d'un événement qui ne provoque pas de changement d'état de l'objet, mais une action comme l'appel d'une opération ou l'émission d'un message. Par exemple, lors d'un mouvement de stock, un article peut ne pas changer d'état si la quantité restant disponible en stock est suffisante et ne passe pas le seuil d'alerte ou de rupture.

Transition locale

Une transition locale s'applique aux sous-états d'un état composite. Elle peut provoquer un changement d'état uniquement à l'intérieur de l'état composite.

Effet d'une transition

Le déclenchement d'une transition peut être accompagné d'un effet. L'effet peut être représenté par :

- Une activité
- Une collaboration
- Une interaction
- Une machine à état

Pour définir l'effet d'une transition :

1. Ouvrez la fenêtre de propriétés de la transition.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Cliquez sur la flèche située à l'extrémité du champ **Effet (Comportement)** et créez ou reliez l'objet qui définit l'effet.

Affichage des effets d'une transition

Pour modifier l'affichage des effets de la transition.

1. Dans le diagramme de machine à état, cliquez avec le bouton droit sur la transition puis cliquez sur **Formes et détails**.
2. Sélectionnez "Effet" dans l'arbre qui s'affiche.

Vous pouvez choisir d'afficher tout ou partie des effets de la transition, avec leurs caractéristiques.

Evénement déclencheur d'une transition

Dans la fenêtre de propriétés d'une transition, sous l'onglet **Evénement**, vous pouvez indiquer le **Type d'Evénement** qui déclenche une transition.

Ce peut être :

- Un événement quelconque
- L'appel d'une opération
- Un changement de l'objet concerné par la transition
- La création d'un objet
- La destruction d'un objet
- L'envoi d'un signal
- L'envoi d'une opération
- L'émission d'un signal par l'objet
- La réception d'un signal
- La réception d'une opération
- Un *temporisateur*

> 🕮 *Un temporisateur est un événement déterminé uniquement par le temps qui s'écoule. Ex : Le lundi, à quatre heures, etc.*

Les champs affichés sous le champ **Type d'événement** varient selon le type d'événement sélectionné.

Vous pouvez sélectionner l'objet concerné par l'effet.

Dans le cas d'une opération ou d'un signal, il est possible de préciser les valeurs des paramètres transmis.

LE DIAGRAMME D'ACTIVITÉS

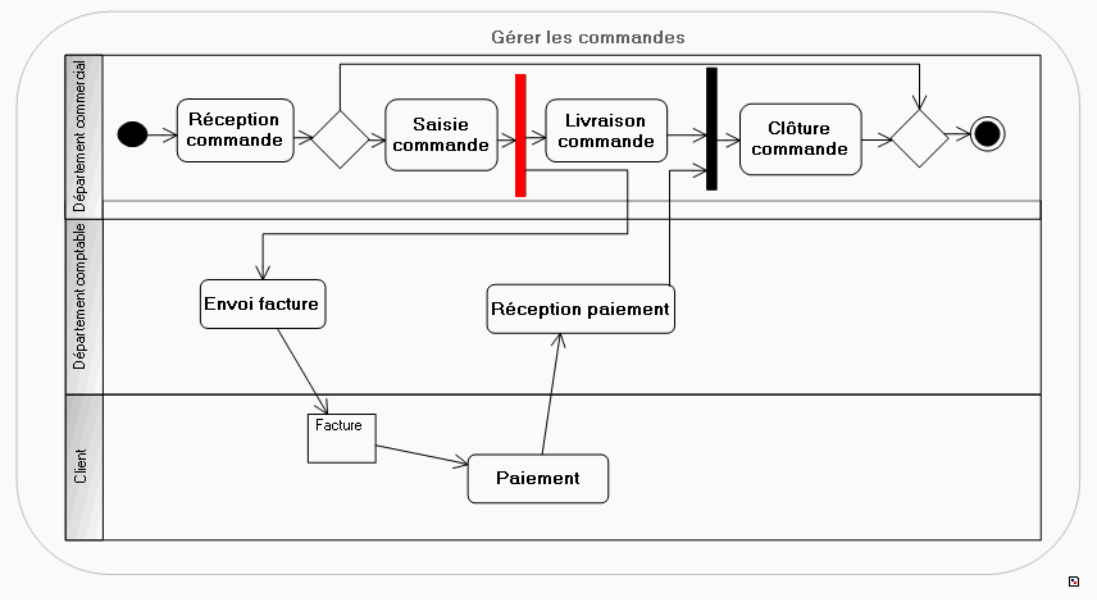
Un diagramme d'activités est proche du diagramme de machine à états. A la différence du diagramme de machine à états qui décrit le comportement d'un objet via l'enchaînement d'états, le diagramme d'activités décrit le comportement d'un élément en termes d'actions.

PRÉSENTATION DU DIAGRAMME D'ACTIVITÉS

Un diagramme d'activités représente un séquencement d'étapes décrivant le comportement d'un élément du système.

Les étapes sont modélisées par des noeuds - noeuds d'action, de paramétrage ou de contrôle - coordonnés par des flux de données ou de contrôle.

Exemple de diagramme d'activités



Créer un diagramme d'activités

Un diagramme d'activités se crée depuis une activité.

Vous pouvez créer une activité depuis un paquetage, un composant ou une classe.

Pour créer un diagramme d'activités :

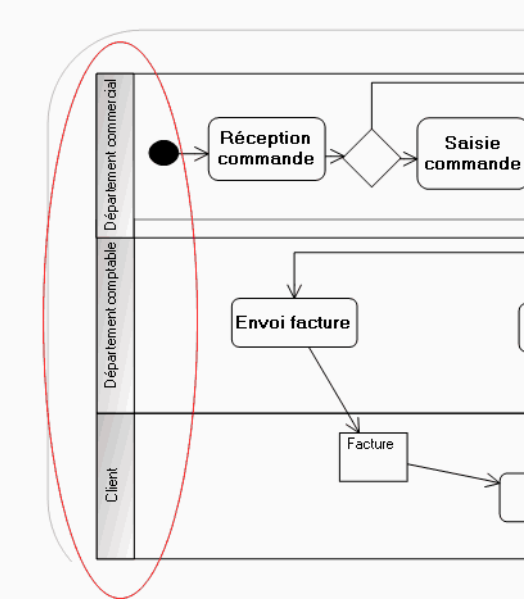
1. Cliquez avec le bouton droit sur l'activité concernée.
2. Dans le menu contextuel qui s'affiche, cliquez sur **Nouveau** > **Diagramme**.
3. Dans la fenêtre qui apparaît, sélectionnez "Diagramme d'activités" et cliquez sur le bouton **Créer**.
   Le nouveau diagramme d'activités s'ouvre.

LES PARTITIONS

Un diagramme d'activités peut être découpé en partitions. Chaque partition contient des nœuds ou des actions ainsi que les flux entre ces éléments.

Vous pouvez utiliser des partitions pour organiser les tâches ou pour spécifier l'élément responsable de la mise en œuvre d'un ensemble de tâches.

Pour plus de détails sur les couloirs, voir le guide **HOPEX Common Features**, chapitre "Manipuler les diagrammes", section "Utiliser les couloirs".



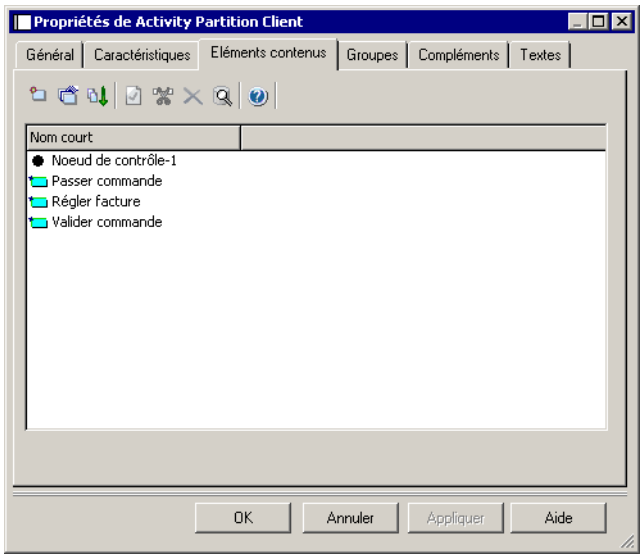
Créer une partition

Pour créer une partition dans le diagramme d'activités :

1. Cliquez sur le bouton **Partition** de la barre d'insertion d'objets.
2. Indiquez son nom.
3. Indiquez l'élément représenté par la partition. Il s'agit de l'élément qui met en oeuvre les éléments de la partition. Ce peut être un acteur, une classe ou un composant.
4. Cliquez sur **OK**.

Propriétés d’une partition

L'onglet **Eléments contenus** de la fenêtre de propriétés de la partition présente les éléments exécutés dans la partition.



L’onglet **Compléments** permet de rattacher la partition aux contraintes qui y sont mises en oeuvre.

LES NOEUDS

Les nœuds permettent de modéliser les étapes de l'activité. Il existe différents types de nœuds dans **HOPEX**.

Les nœuds d'actions

Les actions sont les étapes élémentaires du comportement représenté par l'activité.

La coordination des actions est réalisée à l'aide de flux de contrôle et de flux de données.

Créer une action

Pour créer une action :

1. Sélectionnez le bouton correspondant au type de l'action dans la barre d'insertion d'objets du diagramme, puis cliquez sur le plan de travail.
   La fenêtre d'ajout d'une action du type choisi s'ouvre.
2. Indiquez son nom et cliquez sur **Créer**.

Types d'action

Dans la fenêtre de propriétés de l'action, sous l'onglet **Caractéristiques**, vous pouvez préciser le type de l'action. Ce peut être :

- L'appel d'une opération d'un autre objet
- La création d'un objet
- La destruction d'un objet
- L'exécution d'une opération locale à l'objet
- L'émission d'un signal par l'objet
- La destruction finale de l'objet
- Etc.

Les nœuds de paramétrage

Les noeuds de paramètrage d'une activité décrivent les entrées ou les sorties de cette activité.

Ils transmettent les paramètres à l'activité par l'intermédiaire des flux qu'ils émettent ou qu'ils reçoivent.

Les nœuds de contrôle

Un noeud de contrôle coordonne les flux entre les noeuds d'une activité.

Un noeud de contrôle peut être de type initial, final, décision, fusion (merge), parallélisme (fork) ou union (join).

Types de noeud de contrôle

Initial

Un noeud initial indique où débute le flux de contrôle lorsque l'activité est invoquée. Une activité peut avoir plusieurs noeuds initiaux.

Final

Lorsqu'un jeton atteint un noeud final d'activité, tous les flux de l'activité sont stoppés. Au contraire, un noeud final de flux détruit les jetons qui lui arrivent mais n'a aucun effet sur les autres jetons de l'activité.

Décision

Une décision fait le choix d'un seul flux entre plusieurs flux sortants possibles. Les flux sortants sont sélectionnés en fonction de leur condition de garde.

Fusion

Une fusion (merge) rassemble plusieurs flux alternatifs entrants en un seul flux sortant. Elle n'est pas utilisée pour synchroniser des flux concurrents mais pour accepter un seul flux parmi plusieurs.

Parallélisme

Un parallélisme (fork) sépare un flux en plusieurs flux concurrents. Les jetons arrivant à un parallélisme sont dupliqués à travers les flux sortants.

Union

Une union (join) synchronise des flux multiples. Quand tous les flux en entrée sont disponibles, le flux en sortie est déclenché.

Les nœuds d'objets : pins d'entrée, de sortie et d'échange

Pour spécifier les valeurs en entrée d'une action et les valeurs de retour, on utilise des nœuds d'objets appelés pins (pin en anglais) d'entrée ou de sortie. L'action ne peut débuter que si une valeur est affectée au pin d'entrée. De même, quand l'action se termine, une valeur doit être affectée au pin de sortie.

Pin d'entrée

Un pin d'entrée supporte les valeurs d'entrée qui doivent être consommées par une action et qu'il reçoit de la part d'autres actions.

Pin de sortie

Un pin de sortie supporte les valeurs de sortie qui sont produites par une action et fournit ces valeurs à d'autres actions à travers des flux.

Pin d'échange

Un pin d'échange est utilisé pour représenter les données échangées entre deux actions.

Les Flux

Le passage d'un nœud à un autre est matérialisé par un flux.

Flux de contrôle

Un flux de contrôle démarre un noeud d'action lorsque le précédent est terminé. Les objets et les données ne peuvent pas être transmis par un flux de contrôle.

Flux d'objets

Un flux d'objets permet de transmettre des données ou objets d'un noeud à un autre à l'intérieur d'une activité.

LES DIAGRAMMES D'INTERACTION

Les diagrammes d'interaction, c'est-à-dire le diagramme de séquence, le diagramme de communication et le diagramme de vue générale d'interaction, représentent une série d'interactions entre objets, ordonnée dans le temps. Ils montrent une ou plusieurs histoires possibles du système.

Les points suivants sont abordés ici : :

LES INTERACTIONS

Une interaction décrit le comportement d'un système dans un contexte particulier par les échanges de messages entre les éléments de ce système.

Quand les diagrammes de machines à état ou d'activités étudient des comportements individuels, les diagrammes d'interaction se concentrent sur la coopération d'un ensemble d'objets.

Créer une interaction

Vous pouvez créer une interaction depuis :

- Un composant
- Un paquetage
- Une classe

Créer un diagramme d'interaction

Le diagramme de séquence, le diagramme de communication et le diagramme de vue générale d'interaction se crée depuis une interaction.

Pour créer un diagramme d'interaction :

1. Cliquez avec le bouton droit sur une interaction.
2. Dans le menu contextuel qui s'affiche, cliquez sur **Nouveau** > **Diagramme**.
   Une fenêtre contenant les différents types de diagramme possibles apparaît.
3. Sélectionnez le type de diagramme voulu et cliquez sur le bouton **Créer**.

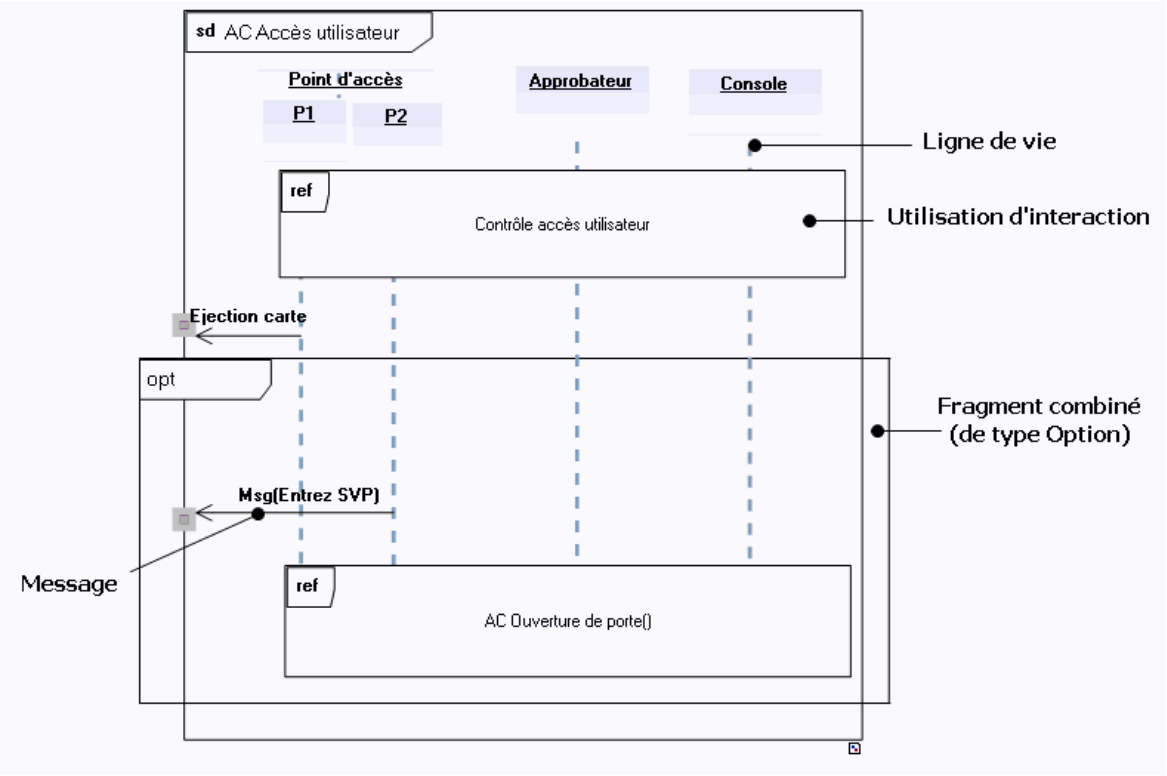
LE DIAGRAMME DE SÉQUENCE

Le diagramme de séquence met en évidence la chronologie des messages échangés entre les objets participant à une interaction. Ces objets sont représentés dans le diagramme par leurs lignes de vie.

Exemple de diagramme de séquence

Le diagramme ci-dessous décrit le comportement d'un distributeur automatique :

- Deux points d'entrée (représentés par des lignes de vie) donnent lieu à un contrôle d'accès de l'utilisateur. Ce contrôle est décrit dans une interaction.
- Selon le résultat du contrôle, l'accès est refusé et la carte de l'utilisateur éjectée ou l'ouverture de la porte est actionnée.
- Un comportement optionnel (représenté par un fragment combiné) peut influencer l'ouverture de la porte.



Les lignes de vie

Une ligne de vie représente un participant dans une interaction.

Les lignes de vie sont des instances de différents types (de classe, d'acteurs, etc.). La flèche située à droite du bouton de ligne de vie offre un raccourci vers les types d'objets Classe ou Composant, plus fréquemment utilisés.

Dans un diagramme de séquence, le temps est représenté comme s'écoulant du haut vers le bas le long des lignes de vie de ces objets. Entre ces objets transitent des instances de messages.

☛ *Les instances représentées dans un diagramme de séquence peuvent être des instances de classe, d'acteur, de paquetage, de cas d'utilisation, de composant ou de nœud, ce qui permet de définir des diagrammes de séquence au niveau de détail souhaité.*

Créer une ligne de vie

Pour créer une ligne de vie :

1. Cliquez sur le bouton **Ligne de vie**  .
2. Cliquez dans le diagramme.
   Une fenêtre s'ouvre.
3. Saisissez le nom de la ligne de vie.
4. Indiquez l'élément représenté par la ligne de vie.
5. Cliquez sur **Créer**.
   La ligne de vie apparaît dans le diagramme.

Pour insérer plusieurs objets existants sur le diagramme en une fois :

1. Cliquez successivement sur le bouton de ligne de vie et sur le bouton Rechercher 
2. Dans la fenêtre qui apparaît, cliquez sur **chercher**.
3. En maintenant la touche <CTRL> enfoncée, sélectionnez dans la liste les objets qui vous intéressent et glissez-les dans le diagramme.

Propriétés d'une ligne de vie

Pour accéder aux propriétés d'une ligne de vie :

- Cliquez avec le bouton droit sur l'instance et sélectionnez **Propriétés**.

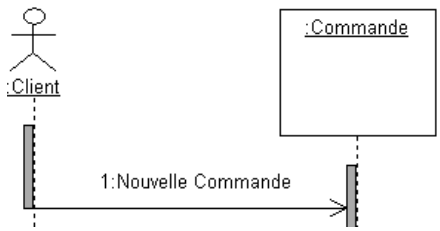
Vous pouvez sélectionner le **Type** de l'objet (Acteur, Classe, etc.), préciser de quelle **Classe**, **Acteur,** etc. cet objet est une instance et indiquer son *stéréotype*.

Les messages

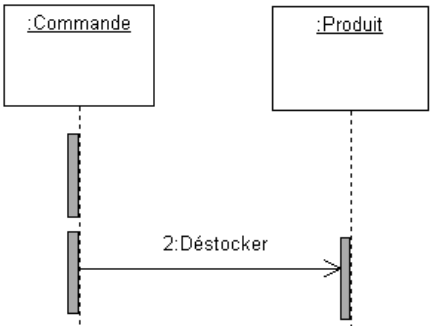
Un message définit une communication particulière entre les lignes de vie d'une interaction. Il spécifie l'émetteur et le récepteur par l'intermédiaire de spécifications d'occurrence, ainsi que le type de communication. Cette communication peut être, par exemple, l'émission d'un signal, l'invocation d'une opération, la création ou la destruction d'une instance.

Exemples de messages échangés

1) Le message envoyé par l'acteur "Client" à la classe "Commande" transporte le signal "Nouvelle commande".



2) Le message envoyé par la classe "Commande" à la classe "Produit" appelle l'opération "Déstocker".



Créer un message

Pour créer un message dans le diagramme de séquence :

1. Cliquez sur le bouton **Message** de la barre d'insertion d'objets en sélectionnant le type de message voulu.

2. Allez de la ligne pointillée sous le premier objet à celle qui est sous le deuxième objet en maintenant le bouton gauche de la souris enfoncé. Le message échangé entre les deux objets se dessine.

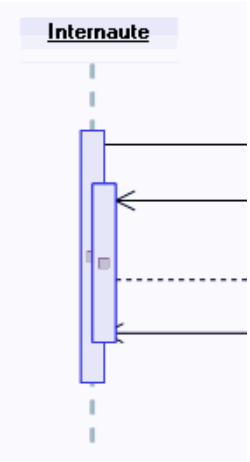
Types de messages

Vous pouvez créer quatre types de messages :

- Dans un message de type "Complet", l'émetteur et le destinataire sont tous les deux définis.
- Dans un message "Perdu", seul l'émetteur est connu. On considère ici que le message n'atteint jamais sa destination.
- Dans un message "Trouvé", seul le destinataire est connu. C'est le cas lorsque l'origine du message se situe en dehors du contexte de description.
- Dans un message de type "Inconnu", ni l'émetteur ni le destinataire ne sont définis.

Occurrence d'exécution

Une occurrence d'exécution d'une ligne de vie (execution specification) représente une unité d'action ou de comportement qui se déroule à partir d'une occurrence d'événement de début jusqu'à une occurrence d'événement de fin.



Créer une occurrence d'exécution

Pour créer une occurrence d'exécution :

1. Dans le diagramme de séquence, cliquez sur le bouton **Occurrence d'exécution** de la barre d'insertion d'objets.
2. Positionnez-la sur la ligne de vie concernée.
   L'occurrence apparaît dans le diagramme.

Occurrence d'événement

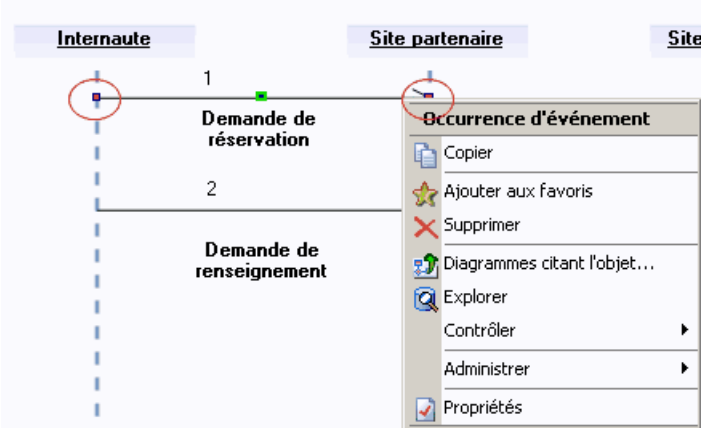
La création d'un message ou d'une occurrence d'exécution entraîne automatiquement la création d'occurrences d'événements.

Une occurrence d'événement (Occurrence Specification) est un point syntaxique à l'extrémité d'un message ou au début ou à la fin d'une occurrence d'exécution.

Les occurrences d'événement sont ordonnées le long d'une ligne de vie.

Ce sont les unités sémantiques de base d'une interaction.

Vous pouvez accéder au menu contextuel d'une occurrence d'événement en cliquant avec le bouton droit sur l'une des extrémités d'un message.



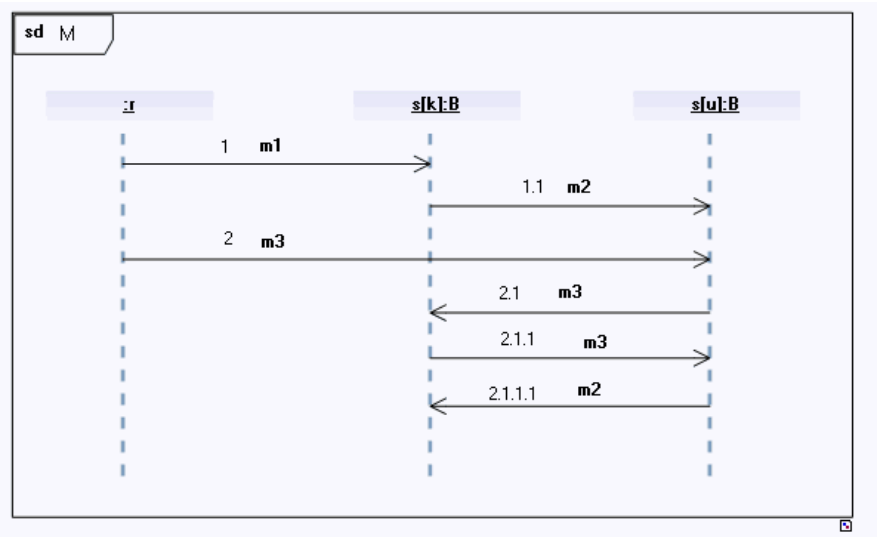
Calcul des numéros de séquence

A partir du positionnement des occurrences d'événement, un outil de calcul permet d'ordonner les messages et les occurrences d'exécution.

Pour ordonner les messages circulant entre des lignes de vie d'une interaction :

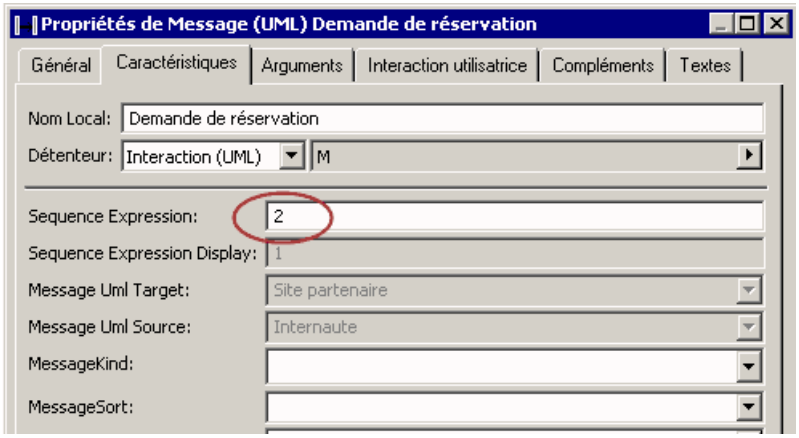
1. Ouvrez le menu contextuel de l'interaction décrite.
2. Cliquez sur **Calculer les numéros de séquence**.
   L'outil applique automatiquement des numéros aux messages.

Exemple



Vous pouvez modifier manuellement le numéro de séquence d'un message dans la fenêtre de propriétés du message :

- Cliquez sur l'onglet **Caractéristiques** et modifiez la valeur du champ **Sequence Expression**.



Lorsque vous relancez le calcul des numéros de séquence, ce dernier met à jour le séquencement en fonction des modifications apportées.

Fragment combiné

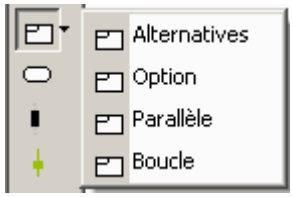
Un fragment combiné permet de décrire de manière concise plusieurs séquences d'exécution.

Un fragment combiné est défini par un opérateur d'interaction et les opérandes d'interactions correspondants.

Créer un fragment combiné

Pour créer un fragment combiné :

1. Dans la barre d'insertion d'objets du diagramme de séquence, cliquez sur le bouton **Fragment combiné**.
   Vous pouvez associer au fragment combiné différents types d'opérateur d'interaction. La flèche située à droite du bouton offre un raccourci vers quatre d'entre eux. Voir "Type d'opérateur d'interaction", page 132.

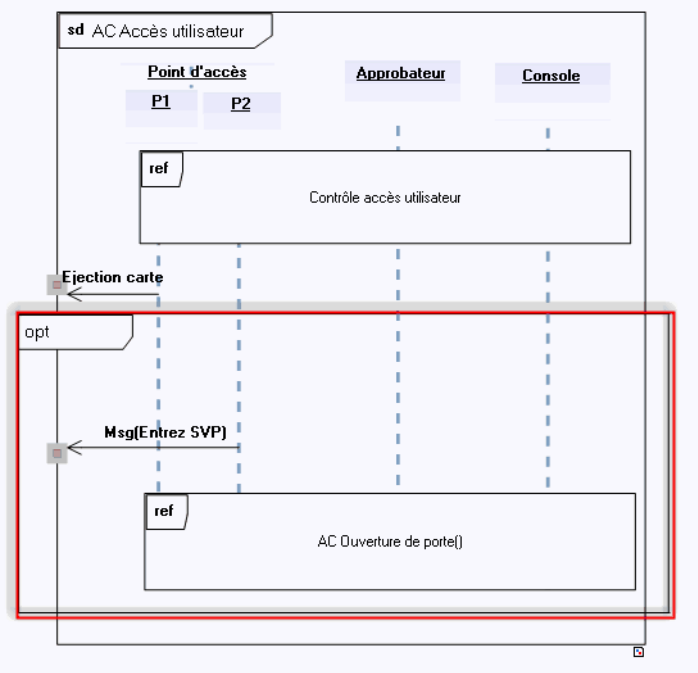


2. Cliquez dans le diagramme.
   La fenêtre de création du fragment combiné apparaît.

3. Indiquez son **Nom** et le **Type d'opérateur d'interaction** si ce n'est pas déjà fait.
4. Cliquez sur **Terminer**.

Un fragment combiné est représenté par un rectangle dont l'angle supérieur gauche affiche le type d'opérateur d'interaction.

Dans l'exemple ci-dessous, un fragment combiné de type optionnel traduit un comportement susceptible de contrarier le séquencement normal (l'ouverture de la porte).



Type d'opérateur d'interaction

Le type d'opérateur d'interaction conditionne la signification du fragment combiné. Il existe différents types d'opérateurs : seq, alt, opt, break, par, strict, loop, region, neg, assert, ignore et consider.

Alternatives

Alt exprime la possibilité de choisir entre différents comportements possibles en évaluant les conditions de garde associées à chacun des opérandes. Au plus un des opérandes pourra être exécuté.

L'opérande gardé par Else est choisi lorsqu'aucune des autres conditions n'est réalisée.

Option

Opt représente un choix entre l'unique opérande proposé ou aucun.

Arrêt (Break)

Break représente un scénario d'arrêt qui est exécuté à la place du reste du fragment d'interaction englobant.

Parallèle

Par implique que les différents opérandes peuvent être exécutés en parallèle. Les occurrences des événements des divers opérandes d'interaction peuvent être entrelacées de toutes les façons tant que l'ordre imposé par chaque opérande est préservé.

Séquence faible (Weak Sequencing)

Seq désigne un entrelacement faible entre les comportements des opérandes défini par trois propriétés :

- L'ordre des occurrences d'événements à l'intérieur de chacun des opérandes est maintenu dans le résultat.

- Les occurrences d'événements de différentes lignes de vie venant de différents opérandes peuvent apparaître dans n'importe quel ordre.

- Les occurrences d'événements d'une même ligne de vie venant de différents opérandes sont ordonnés de telle sorte que l'occurrence d'événement du premier opérande apparaît avant celle du deuxième.

Séquence stricte (Strict Sequencing)

Strict définit un séquencement strict entre les comportements des opérandes.

Négation (Negative)

Neg représente un opérande invalide.

Région critique

Critical représente une région qui doit être traitée de manière atomique, ce qui signifie que des occurrences d'événement ne peuvent pas être entrelacées avec celles de la région critique.

Ignorer / Considérer

Consider et Ignore nécessitent qu'une liste de messages pertinents soient spécifiés.

Ignorer indique que les types de certains messages sont ignorés dans le fragment combiné.

Consider signifie que seuls certains messages vont être considérés à l'intérieur du fragment combiné. C'est équivalent à définir tous les autres messages comme 'ignorés'.

Assertion

Assert représente une séquence qui est la seule valide pour un message donné.

Ainsi, toute séquence définie par un fragment d'interaction qui commence par les messages qui aboutissent à la séquence définie par le bloc assert et qui continue par un échange de messages ne respectant pas le bloc assert doit être définie comme invalide.

Les assertions sont fréquemment utilisées en combinaison avec les types Ignore et Consider.

Boucle

Loop permet d'indiquer que l'opérande d'interaction sera répété un certain nombre de fois. Il est possible de spécifier un nombre minimum et un nombre maximum de boucles, ainsi qu'une expression de continuation de la boucle.

Opérande d'interaction

Un opérande d'interaction est contenu dans un fragment combiné et représente un opérande de l'expression donnée par le fragment combiné englobant. Il peut être conditionné par une contrainte d'interaction qui sert de condition de garde.

Créer un opérande d'interaction

Pour créer un opérande d'interaction :

1. Faites un clic droit sur le fragment combiné qui contient l'opérande d'interaction.
2. Sélectionnez **Nouveau** > **Opérande d'interaction**.
3. Nommez l'opérande et cliquez sur **OK**.

Créer une contrainte d'interaction

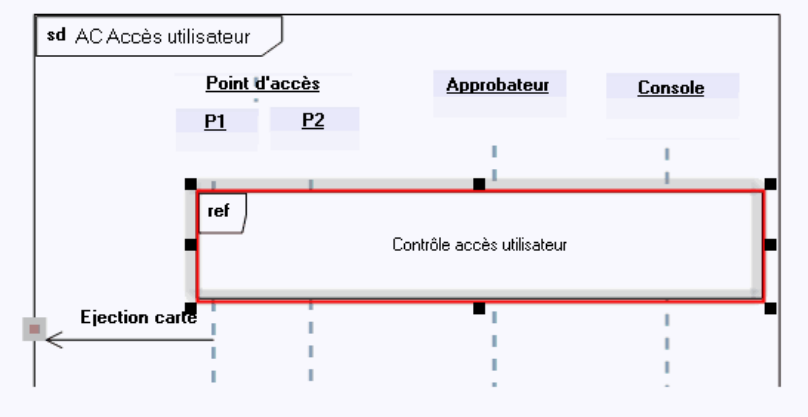
Pour créer la contrainte d'interaction qui va conditionner l'opérande :

1. Ouvrez la fenêtre de propriétés de l'opérande d'interaction.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Dans le cadre **Condition**, cliquez sur **Nouveau**.
4. La condition est représentée par une contrainte. Définissez la contrainte et cliquez sur **OK**.

Utilisation d'interaction

Une utilisation d'interaction se réfère à une interaction. C'est un moyen de copier le contenu de l'interaction référencée à l'endroit de l'occurrence d'interaction.

Exemple



Pour créer une utilisation d'interaction :

1. Cliquez sur le bouton **Utilisation d'interaction**  .
2. Cliquez dans le diagramme.
3. Dans la fenêtre qui apparaît, indiquez son nom et l'interaction appelée.
4. Cliquez sur **Terminer**.

Vous pouvez préciser les arguments d'une utilisation d'interaction. Un argument est une valeur spécifique correspondant à un paramètre de l'interaction appelée. Aussi, une fois l'argument créé sur l'utilisation d'interaction, vous devez le mettre en correspondance avec le paramètre de l'interaction appelée.

Pour créer un argument :

1. Ouvrez la fenêtre de propriétés de l'utilisation d'interaction.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Dans le cadre **Arguments**, cliquez sur le bouton **Nouveau**.
   Une spécification de valeur est créée.

   Vous pouvez la renommer et préciser ses caractéristiques en ouvrant sa fenêtre de propriétés.

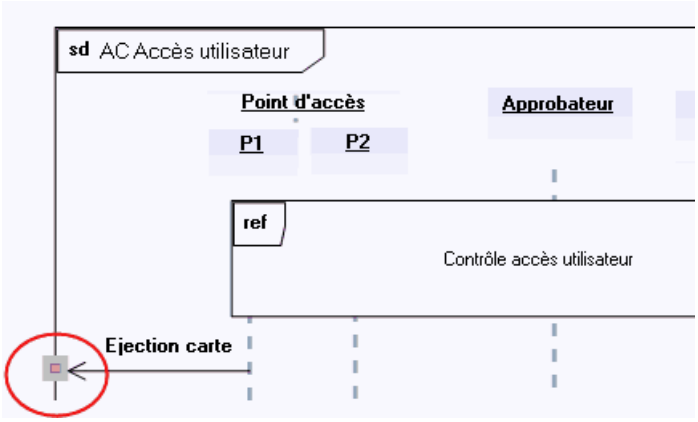
Pour mettre l'argument en correspondance avec le paramètre de l'interaction appelée :

1. Dans la fenêtre de propriétés de l'utilisation d'interaction, cliquez sur l'onglet **Caractéristiques**.
2. Cliquez sur la flèche située à l'extrémité du champ **Interaction called** et sélectionnez **Modifier**.
   Une fenêtre affiche les paramètres de l'interaction appelée.
3. Pour chaque paramètre, cliquez dans la colonne valeur et sélectionnez la spécification de valeur qui lui correspond.

Porte

Une porte (gate) est un point de connexion entre un message extérieur à un fragment d'interaction et un message appartenant à ce fragment d'interaction.

Exemple



Pour créer une porte dans le diagramme de séquence :

1. Cliquez sur le bouton **Porte** de la barre d'insertion d'objets.
2. Cliquez sur le cadre délimitant l'interaction, là où vous souhaitez positionner la porte.
   La porte apparaît dans le diagramme.

Continuation

Une continuation est un moyen syntaxique de définir le prolongement des séquences de différentes branches de fragment combiné alternatif. Les continuations sont similaires à des labels représentant des points intermédiaires dans un flot de contrôle.

LE DIAGRAMME DE COMMUNICATION

Le diagramme de communication est une représentation simplifiée d'un diagramme de séquence ; il se concentre sur les échanges de messages entre les objets au sein d'une interaction.

Le diagramme de séquence et le diagramme de communication sont isomorphes. Lorsqu'un diagramme de communication porte sur une interaction déjà décrite dans un diagramme de séquence, il est automatiquement initialisé à partir des informations contenues dans le diagramme de séquence.

Exemple

Diagramme de séquence

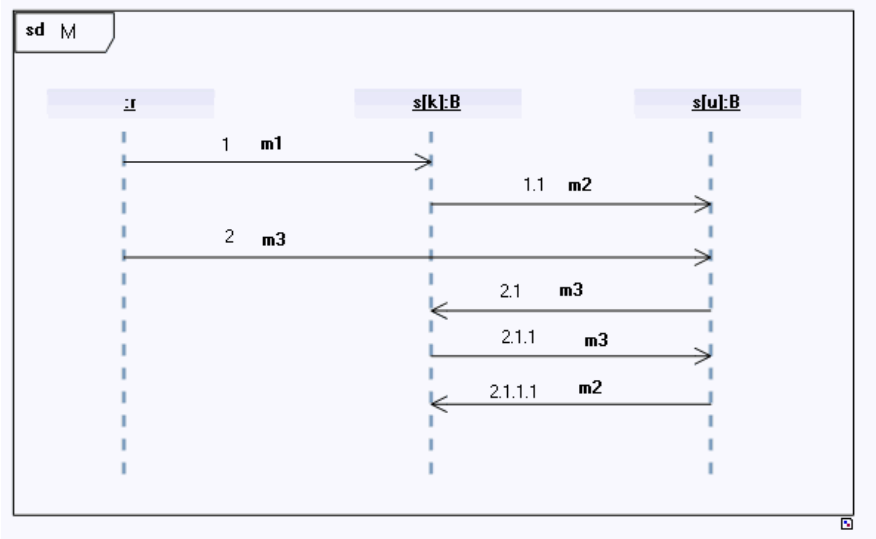
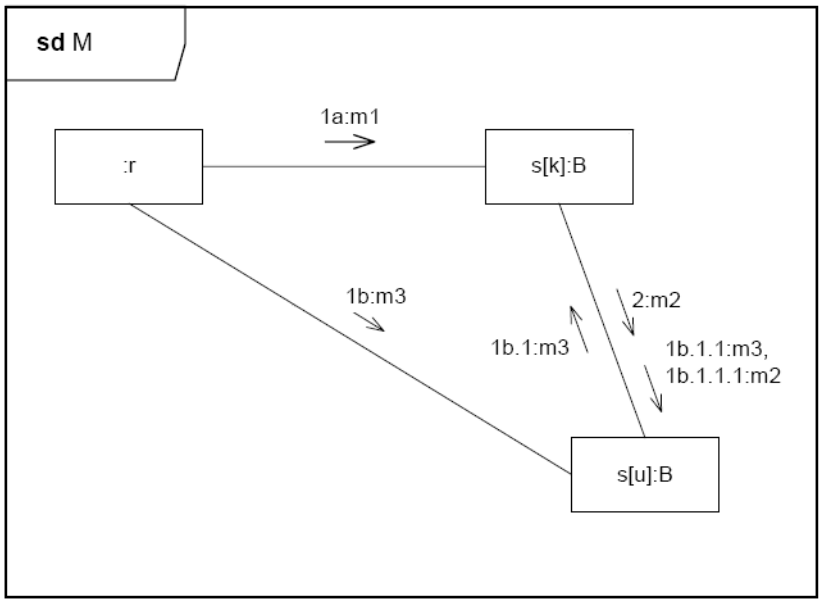


Diagramme de communication

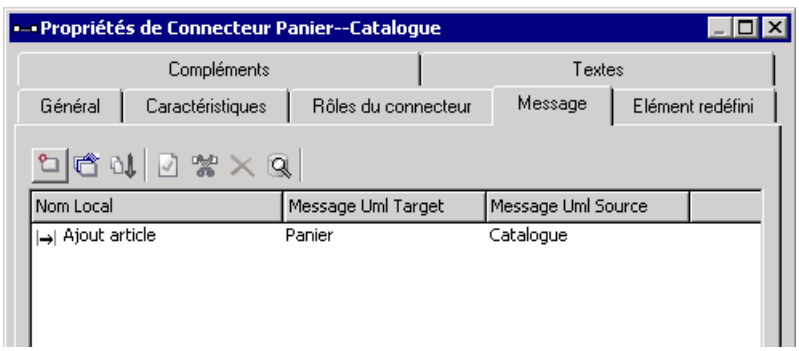


Objets du diagramme

Les objets du diagramme de communication sont les lignes de vie et les messages transmis par des connecteurs.

Lorsque vous reliez deux lignes de vie par un connecteur  , la boîte de création du connecteur propose les messages susceptibles d'être transmis.

Une fois le connecteur créé, vous pouvez également lui associer de nouveaux messages via sa fenêtre de propriétés, sous l'onglet **Message**.

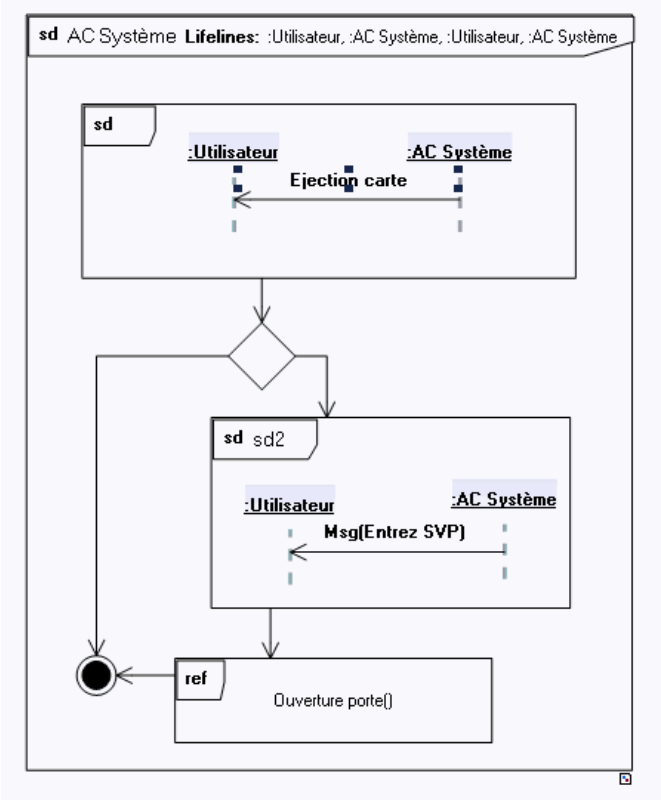


La séquence des messages est donnée par un numéro de séquence associé aux messages. Voir "Calcul des numéros de séquence", page 130.

Pour plus de détails sur les connecteurs, voir "Les connecteurs", page 97.

LE DIAGRAMME DE VUE GÉNÉRALE D'INTERACTION

Le diagramme de vue générale d'interaction permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences. Il vise à fournir une vue d'ensemble du flux de contrôle.



Les objets représentés dans le diagramme de vue générale d'interaction sont les interactions et utilisations d'interaction, les lignes de vie, les messages, les noeuds de contrôle et les flux de contrôle.

LE DIAGRAMME DE DÉPLOIEMENT

Le diagramme de déploiement complète le diagramme de composants en mettant en avant les ressources matérielles sur lesquelles s'exécutent les composants.

✓ "Présentation du diagramme de déploiement", page 142.

PRÉSENTATION DU DIAGRAMME DE DÉPLOIEMENT

Le diagramme de déploiement complète le diagramme de composants. Il décrit les ressources matérielles (ordinateur, routeur etc.) qui composent le système et montre la répartition des composants sur ces matériels.

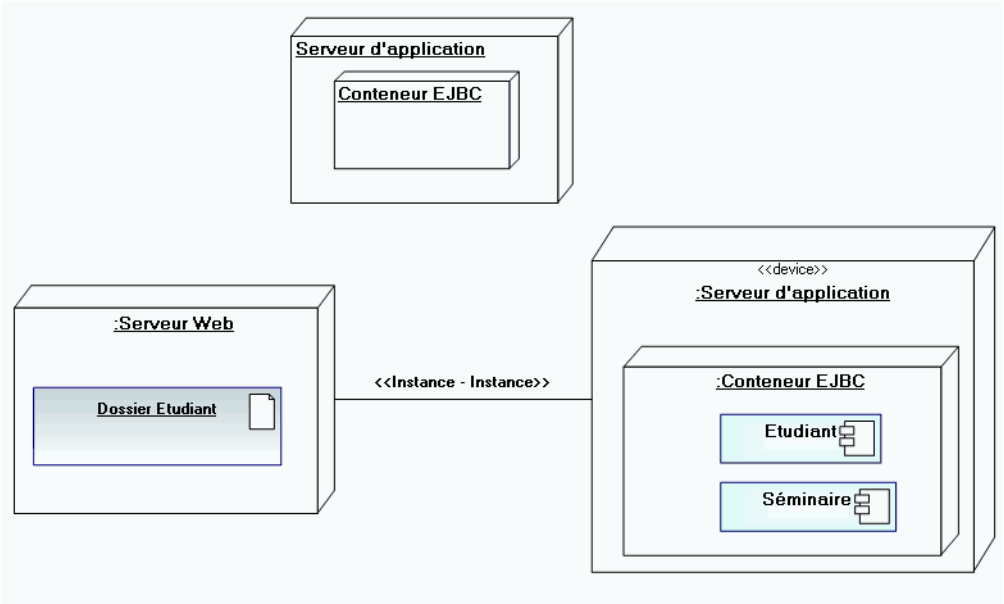
Il décrit également les connexions entre les composants ou les nœuds.

Ce diagramme permet également de préciser les interfaces requises et implémentées pour l'enchaînement des composants.

Il peut être illustré et complété par l'ajout d'instances de noeud, de composant ou de classe.

Vous pouvez créer un diagramme de composants depuis un paquetage.

Exemple de diagramme de déploiement



Objets du diagramme de déploiement

Noeud

Un noeud est un objet physique représentant une ressource informatique disposant généralement d'une mémoire et souvent de capacités de calcul et sur lesquels des composants peuvent être déployés

Les noeuds peuvent se composer d'autre noeuds ou d'artefacts. Pour montrer qu'un composant est affecté à un nœud, il faut soit placer le composant dans le nœud, soit relier le composant au noeud par une relation de dépendance.

Voir "Les liens de dépendance", page 101.

Vous pouvez créer un noeud dans le diagramme de déploiement à l'aide du bouton **Noeud (UML)**  de la barre d'insertion d'objets.

Chemin de communication

Les connexions entre noeuds sont représentées par des chemins de communication par lesquels sont échangés des signaux et des messages.

Composant

Un composant représente une partie modulaire d'un système qui encapsule son contenu et qui est remplaçable dans son environnement. Un composant définit son comportement par les interfaces qu'il fournit et celles qu'il requiert.

Un composant peut être remplacé par un autre si leurs interfaces sont conformes.

Un composant peut être un logiciel, un programme, un élément de code, etc.

Artefact

Un artefact  représente un élément d'information physique qui est utilisé ou produit par le processus de développement d'un logiciel, ou par le déploiement ou la mise en oeuvre d'un système. Ex: fichiers sources, scripts, fichiers binaires exécutables, les livrables issus d'un développement, un document produit par un traitement de texte, un message électronique, etc.

Manifestation

Une manifestation  est la restitution physique concrète dans un artefact d'un ou de plusieurs éléments de modélisation tels que des composants ou des classes.

Une dépendance de manifestation a pour source un artefact et pour cible un composant ou une classe.

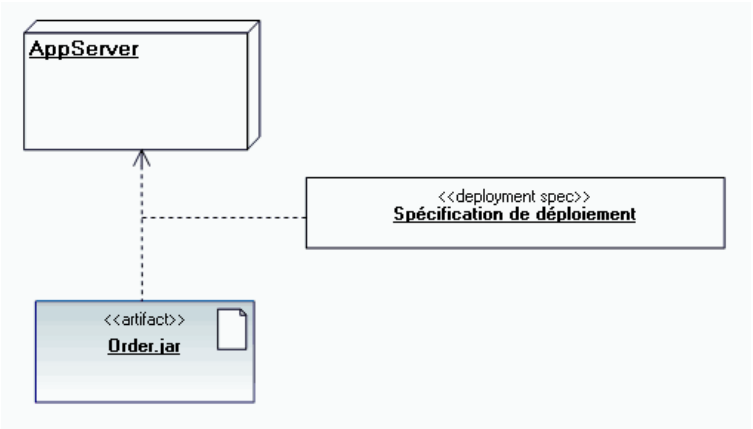
Spécification de déploiement

La spécification d'un déploiement permet d'indiquer l'ensemble des caractéristiques qui déterminent les paramètres d'exécution d'un artefact ou d'un composant déployé sur un noeud.

Configuration

Le bouton Configuration  permet de créer le lien entre une spécification de déploiement et un déploiement.

Exemple



ANNEXE : TYPE DES ATTRIBUTS

Les points suivants sont abordés ici :

TYPES ÉLÉMENTAIRES

Un type élémentaire permet de mettre en commun des caractéristiques communes à plusieurs attributs. Les types élémentaires sont implémentés sous forme de classe.

Définir un type élémentaire

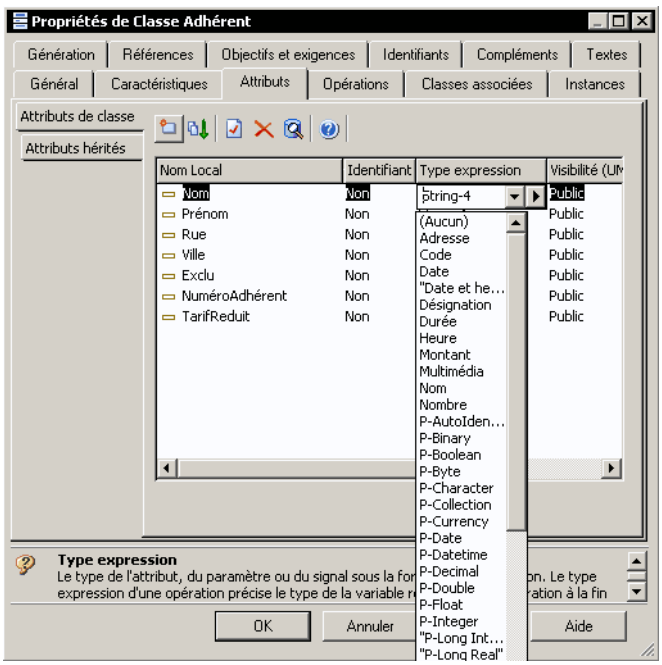
Les types élémentaires sont définis dans un diagramme de classes.

Ce sont des classes pour lesquelles on précise les points suivants :

- Ce sont des classes de stéréotype "Type élémentaire".
- Ce sont des classes "Abstraites" car elles ne sont pas destinées à être instanciées.
- Ce sont des classes "Non persistantes". Elles ne doivent en effet pas donner lieu à une table dans la base de données.

Pour renseigner le type des attributs des classes :

1. Dans la fenêtre de propriétés de la classe, cliquez sur l'onglet **Attributs**.
2. Cliquez dans le champ **Type expression** et sélectionnez le type de l'attribut à l'aide de la flèche.



Les classes proposées en standard sont :

Types alphanumériques		**Compléments**
M-Char	Chaîne de caractères alphanumériques de taille fixe	Longueur
M-Varchar	Chaîne de caractères alphanumériques de taille variable	
Types numériques		
M-Numeric	Numérique	Longueur, Décimale
M-Amount	Montant exprimé en monnaie	Longueur, Décimale
Types dates		
M-Date	Date	
M-Time	Heure	
M-Datetime	Date et heure	
Types binaires		
M-Timestamp	Identification générée automatiquement à partir de la date et de l'heure exprimée en millièmes de secondes après le 01 Janvier 1970	
M-Bool	Booléen valant 0 ou 1	
M-Multimedia	Chaîne binaire	

PAQUETAGES ET TYPES ÉLÉMENTAIRES

Paquetages

Un paquetage partitionne le domaine d'étude et les travaux associés. Il permet de regrouper divers éléments, en particulier des cas d'utilisations et des classes. Un paquetage peut aussi contenir d'autres paquetages. Les paquetages sont liés entre eux à travers des rapports contractuels définissant leur interface.

L'affectation des classes à des paquetages est extrêmement structurante. En effet, comme une classe ne peut être détenue que par un seul paquetage, il est nécessaire de définir des liens clients - fournisseurs entre les paquetages de façon à ce que les paquetages qui en ont besoin puissent utiliser les classes dont ils ne sont pas détenteurs.

Ceci est particulièrement important pour les classes de type élémentaire puisqu'elles vont être utilisées pour définir les attributs des autres classes.

☛ *Règle : Une classe est détenue par un paquetage et un seul.*

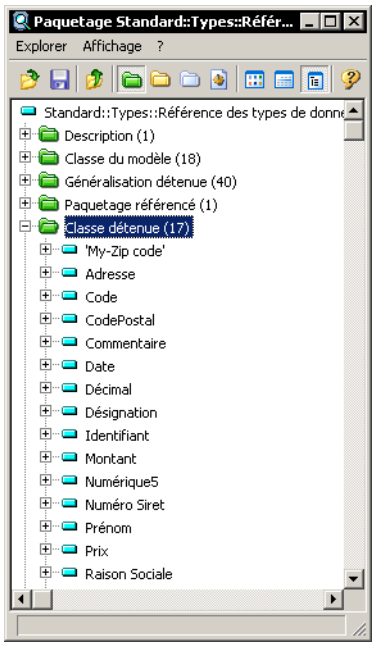
Les types élémentaires disponibles pour les attributs d'une classe dépendent du paquetage qui la détient.

Pour typer les attributs d'une classe, on proposera les types élémentaires définis pour le paquetage qui contient cette classe.

Les types élémentaires disponibles sont les classes publiques de stéréotype "Type élémentaire" détenues ou utilisées par ce paquetage ou par les paquetages dont il est client.

On peut ainsi définir un paquetage de référence (ou plusieurs) détenant les types élémentaires utilisés dans l'entreprise. Chacun des autres paquetages sera déclaré client du paquetage de référence des types élémentaires.

Dans l'exemple ci-dessous, le paquetage "Référence des types de données" détient les classes "Adresse", "Code", "Date", etc.



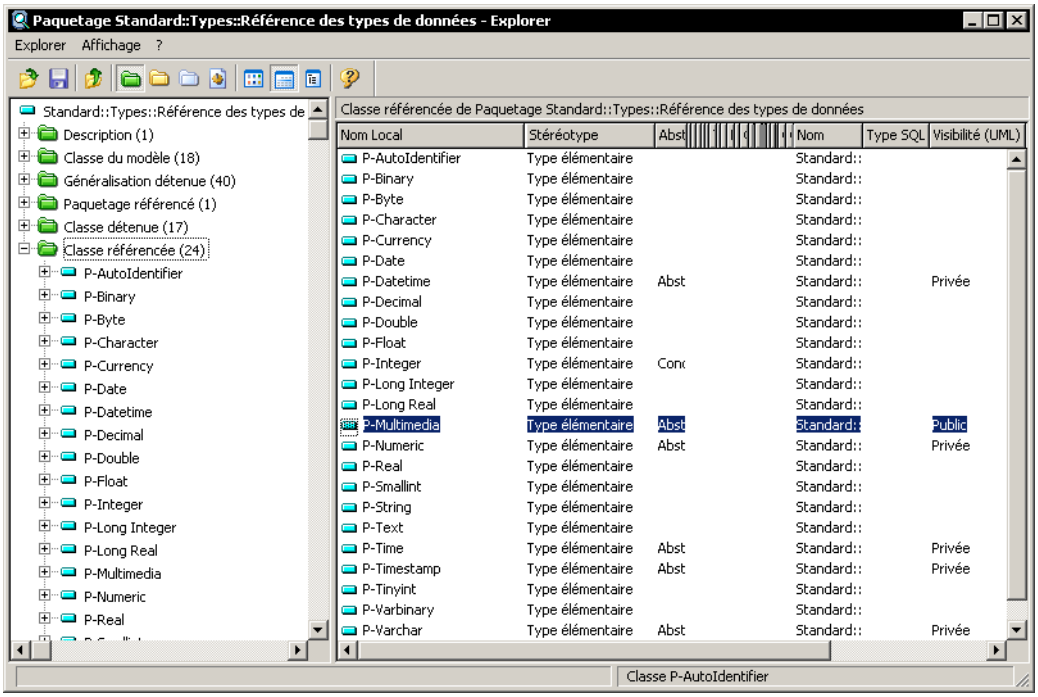
Il est référencé par les paquetages "Bibliothèque", "Gestion des commandes" etc.

Les attributs des classes de ces paquetages peuvent donc être typés à l'aide des types "Adresse", "Code", "Date", etc.

Il est également possible de préciser directement qu'un paquetage référence une classe détenue par un autre paquetage.

Dans l'exemple ci-dessous, les classes "P-Datetime", "P-Multimedia", "P-Numeric", etc., sont référencées par le paquetage "Référence des types de données" sans qu'il les détienne.

Parmi celles-ci, seule la classe "M-Multimedia" est rendue publique par ce paquetage.


Paquetage Standard::Types::Référence des types de données - Explorer
Explorer
Affichage
?
Standard::Types::Référence des types de
Description (1)
Classe du modèle (18)
Généralisation détenue (40)
Paquetage référencé (1)
Classe détenue (17)
Classe référencée (24)
P-AutoIdentifier
P-Binary
P-Byte
P-Character
P-Currency
P-Date
P-Datetime
P-Decimal
P-Double
P-Float
P-Integer
P-Long Integer
P-Long Real
P-Multimedia
P-Numeric
P-Real
Classe référencée de Paquetage Standard::Types::Référence des types de données
Nom Local
Stéréotype
Abst
Nom
Type SQL
Visibilité (UML)
P-AutoIdentifier Type élémentaire Standard::
P-Binary Type élémentaire Standard::
P-Byte Type élémentaire Standard::
P-Character Type élémentaire Standard::
P-Currency Type élémentaire Standard::
P-Date Type élémentaire Standard::
P-Datetime Type élémentaire Abst Standard:: Privée
P-Decimal Type élémentaire Standard::
P-Double Type élémentaire Standard::
P-Float Type élémentaire Standard::
P-Integer Type élémentaire Conc Standard::
P-Long Integer Type élémentaire Standard::
P-Long Real Type élémentaire Standard::
P-Multimedia Type élémentaire Abst Standard:: Public
P-Numeric Type élémentaire Abst Standard:: Privée
P-Real Type élémentaire Standard::
P-Smallint Type élémentaire Standard::
P-String Type élémentaire Standard::
P-Text Type élémentaire Standard::
P-Time Type élémentaire Abst Standard:: Privée
P-Timestamp Type élémentaire Abst Standard:: Privée
P-Tinyint Type élémentaire Standard::
P-Varbinary Type élémentaire Standard::
P-Varchar Type élémentaire Abst Standard:: Privée
Classe P-AutoIdentifier

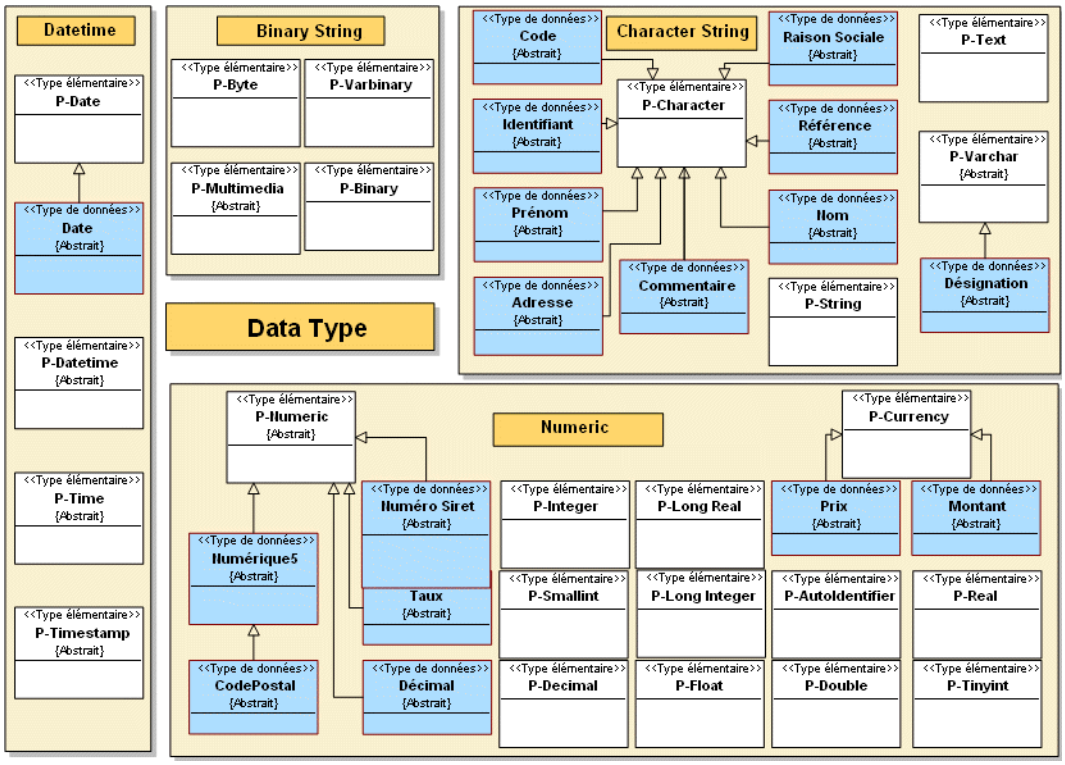
DÉFINIR DE NOUVEAUX TYPES ÉLÉMENTAIRES

De nouveaux types élémentaires peuvent être définis à l'aide d'un diagramme de classes.

Ce diagramme de classes pourra décrire, selon que l'on aura choisi ou non la structuration des classes dans des paquetages :

- Une base de données de référence.
- Le paquetage des types de référence.

Vous pouvez définir vos propres types élémentaires en les déclarant sous-classes des types élémentaires proposés en standard comme dans l'exemple ci-dessous :

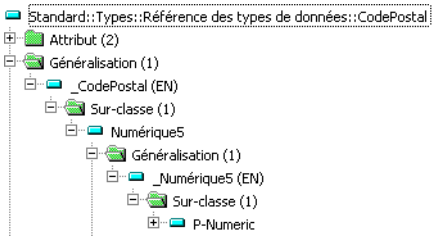


Les types élémentaires définis comme sous-classes vont hériter automatiquement des caractéristiques de leur super-classe. En particulier, la règle de transformation en datatype de la super-classe est appliquée à la sous-classe.

Il est possible de préciser sur la sous-classe une longueur et un nombre de décimales. Ceux-ci seront pris en compte pour la génération des datatypes s'ils n'ont pas déjà été définis pour la super-classe.

L'héritage peut se faire sur plusieurs niveaux.

Dans l'exemple suivant, le type élémentaire "CodePostal" est une spécialisation du type "Numérique5" de longueur 5, lui-même spécialisation du type standard "P-Numeric".

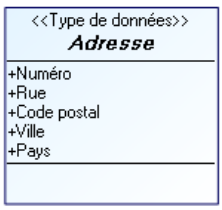


Si le nouveau type élémentaire n'est pas défini directement ou indirectement comme sous-classe d'un type élémentaire standard, il est nécessaire de mettre à jour le tableau de conversion des types élémentaires en datatypes de colonnes.

☛ *Une correspondance peut également être définie directement entre un type et le datatype SQL généré pour chaque SGBD cible sans utiliser le mécanisme d'héritage (voir "Correspondances entre types pivots et datatypes" dans le guide* ***HOPEX Database Builder****).*

Type élémentaire composé

On peut définir un type élémentaire composé en lui précisant une liste d'attributs.

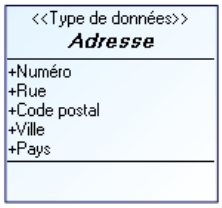


Ici le type Adresse est composé du numéro, de la rue, du code postal, de la ville et du pays.

Un attribut de type Adresse donnera lieu lors de la dérivation à ces cinq colonnes.

Il est possible de décomposer un type à plusieurs niveaux en affectant un type décomposé à l'un de ses attributs.

Par exemple, on peut décomposer le code postal en code ville et code département :



HOPEX XML Schemas

Guide d’utilisation

HOPEX V2

SOMMAIRE

Sommaire

INTRODUCTION

Un schéma XML est un modèle permettant de décrire la structure de l'information contenue dans un document XML.

> 🕮 *XML signifie eXtensible Markup Language. Il s'agit d'un langage de description et d'échange de documents structurés. La première version du langage XML a été adoptée par le W3C en janvier 1998. XML permet d'inventer librement de nouvelles balises. C'est pour cette raison que le langage est dit extensible. Cette extensibilité est à l'origine de tous les apports de XML.*

Grâce à la définition d'une structure de document, il est possible de vérifier la conformité de documents ou de messages avec ce format. Les entreprises partageant les mêmes formats de documents XML peuvent ainsi échanger des documents et messages sur Internet.

HOPEX UML permet de rétro-générer, spécifier et générer des schémas XML.

HOPEX propose deux méthodes pour construire des schémas XML. L'utilisateur peut choisir de travailler :

- Dans l'éditeur XML, où il trouvera les concepts standard XML.
- Dans l'éditeur de diagrammes, où l'utilisateur peut modéliser les schémas XML en utilisant les concepts du standard UML. L'avantage d'utiliser le langage UML est de pouvoir réutiliser des notions telles que l'héritage ou les associations, utiles lors de la modélisation de schémas XML.

Quel que soit le mode utilisé, l'utilisateur spécifie des schémas XML générables au format XSD (eXtensible Schema Definition)

Ce guide explique comment constituer des schémas XML avec **HOPEX UML**, il n'explique pas la méthodologie XML en général.

CONTEXTES D'UTILISATION DES SCHÉMAS XML

Les schémas XML sont utilisés dans les contextes suivants :

- ***La publication d'informations*** fait intervenir de nombreux documents XML ayant des liens complexes.
- ***Traitement des transactions de commerce électronique*** : les bibliothèques de schémas définissent des transactions au sein de places de marchés et entre plusieurs parties.
- ***Acquisition de messages de contrôle et de données*** : la gestion et l'utilisation de périphériques réseaux engendrent l'échange de données et de messages de contrôle. Les schémas peuvent être utilisés par un serveur pour assurer la validité des messages sortants ou par le client pour lui permettre de déterminer quelle partie du message est compréhensible.
- ***Edition de documents traditionnels*** : les schémas peuvent jouer un rôle important dans le cadre des applications guidant les auteurs dans la composition de documents. Ainsi, l'application peut vérifier que les données saisies par l'auteur sont correctes.
- ***Formulations de requêtes*** : il est possible d'utiliser les schémas pour assister et optimiser la formulation de requêtes. Une base de données peut s'auto-définir par un schéma pour informer les autres systèmes de ce qu'elle contient et des requêtes utiles.
- ***Echange de métadonnées*** : l'intérêt pour l'échange de métadonnées est grandissant et les schémas XML permettent de faciliter leur interopérabilité.

> 🕮 *Les métadonnées englobent l'ensemble des informations relatives à la provenance, à l'historique des données de production et à la nature des traitements que celles-ci auront subis pour être exploitables dans un data warehouse.*

CONVENTIONS UTILISÉES DANS CE GUIDE

☛ *Remarque sur les points qui précèdent.*

📖 *Définition des termes employés.*

☺ *Astuce qui peut faciliter la vie de l'utilisateur.*

Compatibilité avec les versions précédentes.

Ce qu'il faut éviter de faire.

Remarque très importante à prendre en compte pour ne pas commettre d'erreurs durant une manipulation.

Les commandes sont présentées ainsi : **Fichier > Ouvrir**.

Les noms de produits et de modules techniques sont présentés ainsi : **HOPEX**.

PRÉSENTATION DE CE GUIDE

Le guide **HOPEX XML Schemas** est composé des chapitres suivants :

- "Présentation de l'éditeur de schémas", page 13 : décrit la fenêtre de l'éditeur.
- "Créer un schéma XML", page 25 : explique comment créer un schéma XML dans **HOPEX UML**.
- "Compléments XSD", page 55 : présente des concepts propres à XSD.
- "Compléments UML", page 65 : explique comment sont gérés les concepts UML tels que l'héritage, les classes associatives et les stéréotypes dans le schéma XML.
- "Contrôles", page 83 : décrit les contrôles liés à la construction des schémas, propres à chaque format de génération.
- "Composants", page 89 : décrit la structure d'un composant et explique comment le générer.
- "Exemples de schémas XML complets", page 93 : fournit des exemples de schéma XML.

  Les chapitres suivants traitent de générations spécifiques :
- "Génération XSD", page 113.
- "Rétro-génération XSD", page 137

PRÉSENTATION

Les concepts XML utilisés dans **HOPEX UML** sont présentés ici, ainsi que l'éditeur de schémas. Avant de construire un schéma XML dans **HOPEX**, il est nécessaire de passer par certaines étapes.

Les points abordés dans ce chapitre sont :

CONDITIONS PRÉALABLES

Afin de permettre au schéma XML de faire référence aux types XSD, il est nécessaire d'importer le Solution Pack "Information Architecture - XsdType".

Importer le Solution Pack XSD

Le Solution Pack "Information Architecture - XsdType" est disponible dans le dossier Utilities\Solution Pack de votre répertoire d'installation HOPEX.

Vous devez le décompresser puis l'importer à partir d'**HOPEX Administration**.

Pour plus de détails, voir "Importer des cadres de référence dans HOPEX", page 240.

DÉFINITIONS

Définition de balise

Une définition de balise permet d'identifier le nom des attributs, la multiplicité des attributs et les sous-balises contenues dans la balise. Ce concept (les classes UML sont utilisées pour définir une définition de balise) définit une famille de balises dont on trouve les instances dans les documents XML. La définition de balise est à distinguer de balise.

Une définition de balise est caractérisée par :

- La liste des *attributs* de la balise.
- Les sous-balises potentiellement imbriquées dans une balise, faisant partie de sa définition.
- Des informations décrivant l'ordre de ces sous-balises, leur fréquence d'apparition, etc.

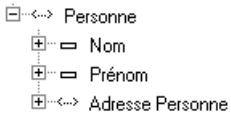
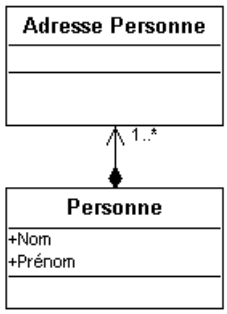
Exemple de définition de balise

```
<xsd:ComplexType name="Personne">
   <xsd:sequence>
      <element name="Adresse_Personne"
type="Adresse_Personne"/>
   </xsd:sequence>
   <xsd :attribute name="Nom" Type="xsd:string"/>
   <xsd :attribute name="Prenom" Type="xsd:string"/>
</xsd:complexType>
```

Exemple d'instance de balise

```
<Personne nom="Smith" prenom="Pierre>
<Adresse_Personne">
   12 rue Cabanis
</Adresse_Personne>
</Personne>
```

Voici les deux représentations possibles dans **HOPEX** de cette définition de balise.

Schéma	Diagrammes
	

Modélisation UML

Une classe UML représente une famille de balises XML et non une balise spécifique d'un document.

☺ *Par le biais des associations, il est possible de réutiliser la définition de balise pour définir plusieurs balises présentant les mêmes caractéristiques dans un document. Par exemple, les balises <AdresseDeTravail> et <AdressePersonnelle> peuvent utiliser la même définition <Adresse>.*

☛ *Dans l'ensemble de cette documentation, la méthode de modélisation dans un diagramme UML sera expliquée en parallèle dans les paragraphes "Modélisation UML".*

Elément

Un élément représente une balise dans l'éditeur de schémas. Les éléments XML d'une balise A sont définis par l'ensemble des balises potentiellement incluses dans A.

Modélisation UML

Lors de la création d'un élément, une relation hiérarchique apparaît entre les définitions de balises, d'où l'utilisation des associations entre classes.

Lorsque vous créez un *élément*, **HOPEX** crée automatiquement l'association entre la classe déclarant l'élément et le type de l'élément.

Ainsi c'est le couple classe + association qui constitue l'élément.

Exemple

Si la définition de balise "Personne" inclut des balises fondées sur la définition de balise "Adresse_Personne", les classes "Personne" et "Adresse_Personne" définissant ces balises sont reliées par une association.

Propriétés de l'élément

Pour afficher les propriétés de l'élément, cliquez avec le bouton droit sur l'élément et sélectionnez :

- **Propriétés du type** pour afficher les propriétés de la classe définissant l'élément.
- **Propriétés du rôle** pour afficher les propriétés du rôle de l'association correspondant à l'élément.

> ☛ *Lorsque l'éditeur est en mode "standard", le menu contextuel ne présente que les propriétés du rôle. Voir "Paramètres d'affichage de l'éditeur de schémas", page 20.*

Attribut

Un attribut XML d'une balise est une paire de chaînes de caractères nom-valeur. La deuxième chaîne est entourée par des guillemets. Les attributs XML servent à caractériser la balise. Exemple : <personne nom = 'Dupont'/>.

Espace de nommage

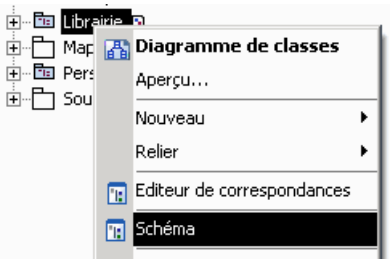
L'espace de nommage est la région d'un vocabulaire (ensemble des définitions de balise, des éléments et des attributs d'un document). A l'intérieur de cet espace, deux types ne peuvent avoir le même nom.

> ☛ *Voir "Accéder aux types de l'espace de nommage", page 21.*

EXEMPLE DE SCHÉMA CRÉÉ DANS HOPEX

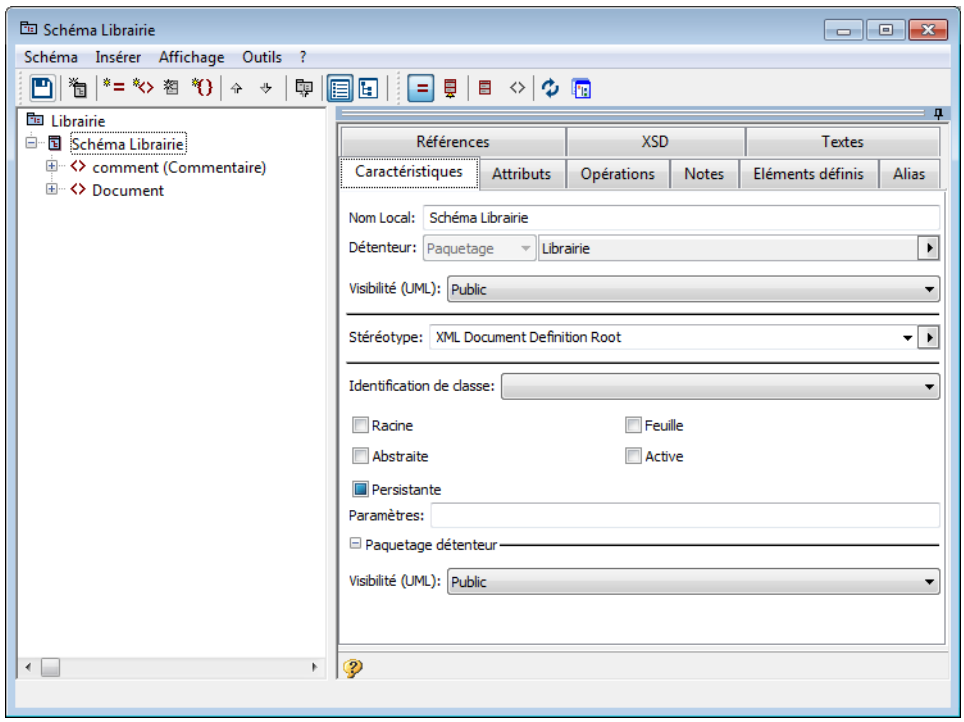
Pour visualiser des exemples de schémas créés dans **HOPEX** :

1. Dans la fenêtre de navigation **Objets principaux**, ouvrez le paquetage "Echange de données".
2. Dans le menu contextuel du paquetage "Librairie", sélectionnez **Schéma**.



L'éditeur de schémas s'ouvre.

3. Ouvrez le schéma en cliquant sur les +.
4. Cliquez sur le bouton **Détails** de la barre d'outils pour afficher la fenêtre de propriétés.



L'exemple présente une structure de document contenant des listes d'auteurs, d'ouvrages et de collections.

La structure du document est représentée par le schéma appelé "Librairie".

Pour préciser le contenu des listes, les auteurs, ouvrages et collections sont inclus en tant qu'"éléments" (ou "sous-balises" <…>) des listes.

Ainsi le schéma est défini comme un arbre hiérarchique où un élément inclut d'autres éléments qui en incluent d'autres.

Des attributs (représentés par des **=**) sont définis au niveau des éléments.

Cette structure de document peut aussi être représentée sous forme de diagramme UML.

Modélisation UML

Pour ouvrir le diagramme UML représentant le schéma "Librairie" :

- Sélectionnez **Diagramme de classes** dans le menu contextuel du paquetage "Librairie".
  Le diagramme de classes s'ouvre.

La représentation hiérarchique est modélisée par les associations composées.

DESCRIPTION DE L'ÉDITEUR DE SCHÉMAS

La partie gauche de l'éditeur fait apparaître un navigateur illustrant la composition des schémas.

La partie droite de l'éditeur fait apparaître la fenêtre de propriétés de l'élément sélectionné dans le navigateur.

Paramètres d'affichage de l'éditeur de schémas

Par défaut, l'éditeur de schémas est en mode standard.

Pour accéder aux fonctionnalités avancées nécessaires à la construction de schémas XML :

- Dans le menu **Outils**, cliquez sur **Niveau expert XML**.
- Dans le menu **Affichage**, cliquez sur **Barre d'outils** et vérifiez que la **Barre de filtrage** est cochée.

Des boutons vous permettent de modifier l'affichage des éléments de votre schéma.

Affichage des attributs : affiche les attributs des éléments.

Affichage des classes héritées : permet de voir les définitions de balise dont une définition de balise a héritées.

Affichage des types : affiche les types de l'espace de nommage.

Affichage des éléments : affiche les éléments.

Conserver l'éditeur au premier plan

 **Arbre seulement** : cache la partie droite de l'éditeur.

 **Détails** : affiche les propriétés des éléments.

 **Actualiser** : rafraîchit la fenêtre pour prendre en compte les nouveaux éléments.

Accéder à la fenêtre de propriétés complète des éléments des schémas

La fenêtre de propriétés de l'éditeur est accessible par le bouton **Détails**  du navigateur. Cette boîte de propriétés est partielle, elle affiche les propriétés relatives à l'élément sélectionné.

Pour accéder à la fenêtre de propriétés complète :

- Cliquez sur le menu **Affichage**.
- Décochez **Filtrage des pages de propriétés**.

Vous pouvez également cliquer avec le bouton droit sur l'élément de votre choix dans le navigateur XML et sélectionner :

- **Propriétés du type :** pour afficher les propriétés de la classe définissant l'élément.
- **Propriétés du rôle** : pour afficher les propriétés du rôle de l'association correspondant à l'élément.

> ☛ *Ces deux commandes apparaissent lorsque vous êtes en mode expert XML.*

Accéder aux types de l'espace de nommage

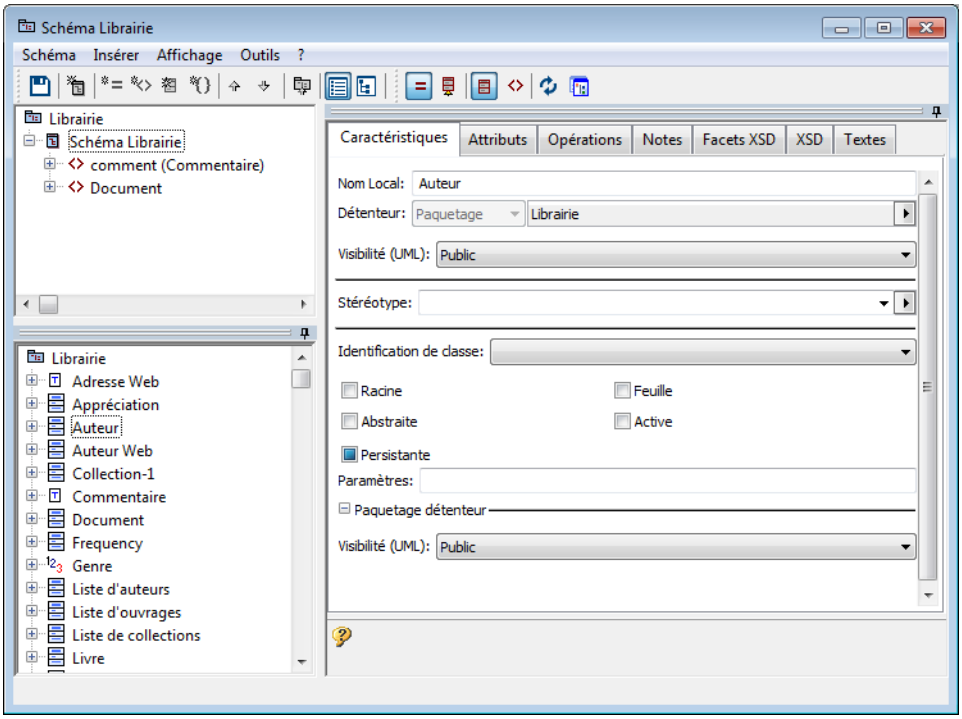
L'éditeur de schémas vous permet d'afficher ou non les types de l'*espace de nommage*.

> 🕮 *L'espace de nommage est la région d'un vocabulaire (ensemble des définitions de balise, des éléments et des attributs d'un document). A l'intérieur de cet espace, deux types ne peuvent avoir le même nom.*

Pour accéder à tous les éléments de l'espace de nommage :

- Cliquez sur le bouton  **Affichage des types**.

Les types s'affichent dans une liste en-dessous du schéma XML.



Lorsque vous cliquez sur un type, sa fenêtre de propriétés apparaît dans la partie droite de l'éditeur.

☛ *Voir aussi "Définir un espace de nommage", page 26.*

Générer un diagramme à partir du schéma XML

Vous pouvez générer un diagramme UML à partir du schéma XML, pour représenter le schéma dans un diagramme.

Pour générer un diagramme à partir du schéma XML :

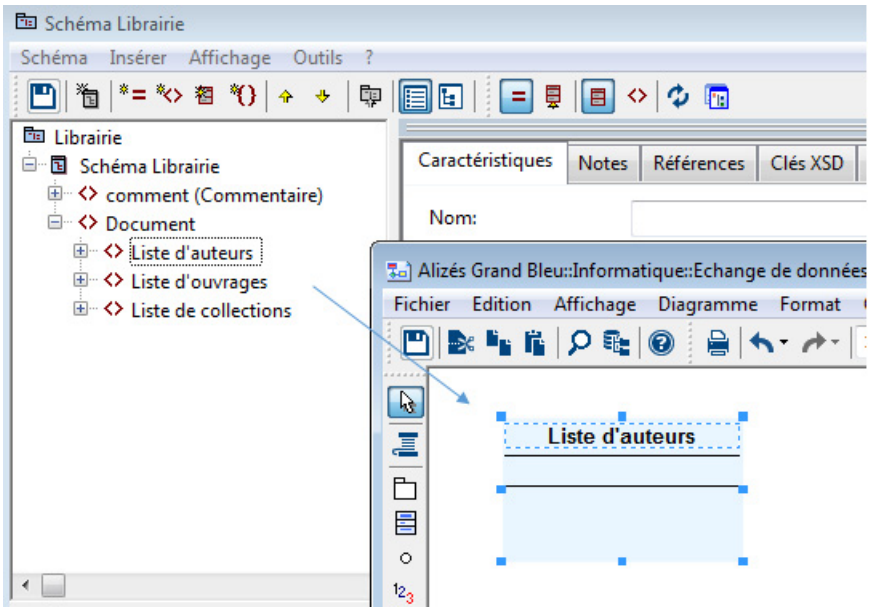
1. Faites un clic droit sur le nom du schéma et sélectionnez **Nouveau** > **Générer un diagramme...**

Vous pouvez générer deux types de diagrammes :

- **Diagramme à 1 niveau** : il décrit pour la classe en question le premier niveau de classes associées avec les éventuelles classes associatives, le premier niveau d'héritage et le premier niveau d'interfaces implémentées.
- **Diagramme multi-niveaux** : il décrit pour la classe en question ***n*** niveaux de classes associées avec les éventuelles classes associatives (la récursion s'arrête si une classe C n'appartient pas au même paquetage que la classe décrite mais si C est incluse dans le diagramme), le premier niveau d'héritage et le premier niveau d'interfaces implémentées.

Vous pouvez également construire vous-même le diagramme.

- Faites un clic droit sur le schéma et sélectionnez **Nouveau** > **Diagramme**.
- Dans la fenêtre qui apparaît; sélectionnez "Diagramme de classes" et cliquez sur **Créer**.
  Le diagramme qui s'affiche est vide.
- Glissez-déplacez les éléments de votre choix du navigateur XML vers le diagramme de classes.



Des boutons vous permettent de créer les classes, associations, etc. représentant les schémas XML. Voir le guide **HOPEX UML** pour les fonctionnalités de la modélisation UML.

A partir de ce diagramme, vous pouvez accéder aux propriétés des éléments du schéma :

- Cliquez avec le bouton droit sur l'objet de votre choix dans le diagramme.
- Sélectionnez **Propriétés**.

La fenêtre de propriétés s'affiche. Vous pouvez spécifier et modifier les propriétés de l'élément.

CRÉER UN SCHÉMA XML

Pour définir la structure d'un document, trois étapes sont nécessaires :

- Créer et nommer le schéma (la structure du document).
- Définir le contenu du document, c'est-à-dire inclure les éléments du schéma.
- Définir les attributs des éléments du schéma.

Les points abordés ici sont :

DÉFINIR UN ESPACE DE NOMMAGE

Afin de permettre au schéma XML de faire référence aux types tels que XSD, il est nécessaire d'importer leur bibliothèque respective. Voir "Importer le Solution Pack XSD", page 14.

Avant de construire un schéma XML, il convient d'abord de définir l'espace de nommage ("namespace") qui constitue l'espace de travail de l'utilisateur. Une fois que l'espace de nommage est créé, vous pouvez ouvrir l'éditeur de schémas XML.

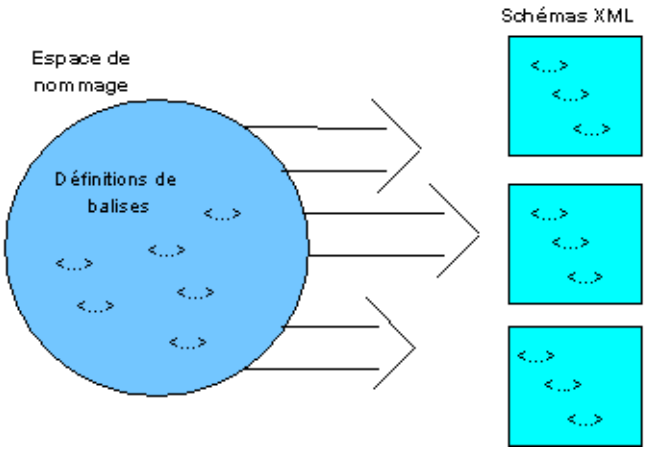
> 🕮 *L'espace de nommage est la région d'un vocabulaire (ensemble des définitions de balise, des éléments et des attributs d'un document). A l'intérieur de cet espace, deux types ne peuvent avoir le même nom.*

Exemple :

```
Dans l'espace de nommage "Bibliothèque", il ne peut y avoir
qu'une seule définition de balise "Ouvrage", réutilisable
plusieurs fois.
```

L'intérêt de définir un *espace de nommage* est de pouvoir réutiliser les types créés à l'intérieur de l'espace de nommage dans un autre espace.

> ☛ *Les types sont préfixés par le nom de l'espace de nommage, ce qui évite des collisions lors de l'importation de ces éléments vers un autre espace de nommage.*

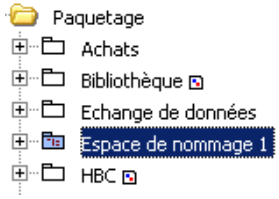


Dans **HOPEX UML**, les espaces de nommage sont représentés par des paquetages.

Pour créer un espace de nommage :

1. Cliquez sur la fenêtre **Objets principaux** du navigateur principal **HOPEX**.
2. Cliquez avec le bouton droit sur **Paquetage**.
3. Sélectionnez **Nouveau** > **Paquetage**.
4. Dans la fenêtre qui s'affiche, saisissez le nom du futur espace de nommage et cliquez sur **OK**.
   Un nouveau paquetage est créé dans la liste des paquetages.

5. Sélectionnez **Propriétés** dans le menu contextuel du nouveau paquetage.
   La fenêtre de propriétés du paquetage s'affiche.
6. Sous l'onglet **Caractéristiques**, dans le champ **Stéréotype**, sélectionnez "XML Document Definition".
7. Cliquez sur **OK**.
   L'espace de nommage est créé dans la liste des paquetages.



Vous pouvez maintenant ouvrir l'éditeur de schémas.

OUVRIR L'ÉDITEUR DE SCHÉMAS

Pour ouvrir l'éditeur de schémas :

- Dans le menu contextuel de l'espace de nommage que vous avez créé, sélectionnez **Schéma**.

L'éditeur de schémas XML s'ouvre. Par défaut, il affiche le "Niveau standard" et filtre un certain nombre de caractéristiques.

Pour accéder aux fonctionnalités avancées nécessaires à la construction de schémas XML :

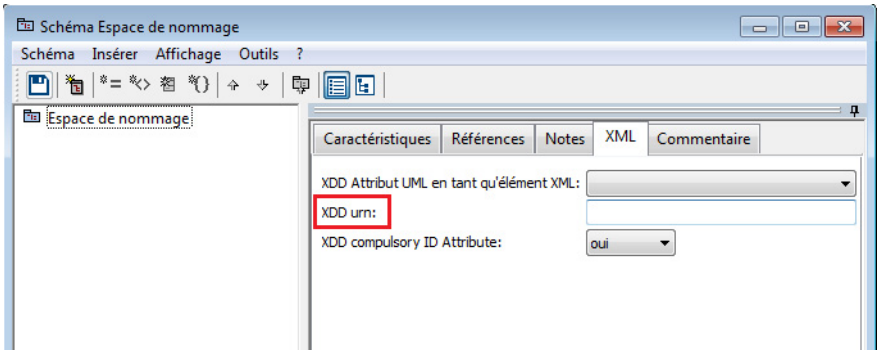
- Dans l'éditeur, cliquez sur le menu **Outils** > **Niveau expert XML**.

Spécifier l'URN de l'espace de nommage

Un URN est l'identifiant technique d'un espace de nommage. Il caractérise l'espace de nommage de manière unique.

Pour spécifier l'*URN* de l'espace de nommage :

1. Dans le navigateur de l'éditeur de schémas, cliquez sur l'espace de nommage.
2. Dans la partie droite de l'éditeur, allez sous l'onglet du langage de génération que vous utilisez et saisissez l'urn du paquetage dans le champ **[XDD] urn**.

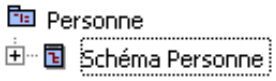


CRÉER UNE CLASSE SCHÉMA

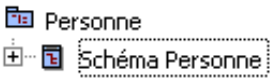
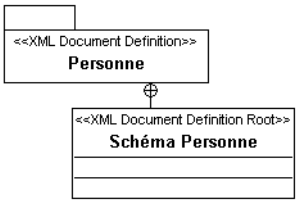
Avant de créer un schéma XML, vous devez avoir défini au préalable un espace de nommage. Voir "Définir un espace de nommage", page 26.

Pour créer un schéma XML :

1. Cliquez sur l'icône **Création d'une classe schéma** dans la barre d'outils de l'éditeur de schémas XML.
2. Dans la fenêtre qui s'affiche, saisissez le nom du schéma et cliquez sur **OK**.
   Le nouveau schéma s'affiche dans le navigateur.



Résultat

Schéma	Diagramme
	

Modélisation UML

Pour représenter un nouveau schéma XML :

1. Cliquez sur le bouton **Classe** et nommez la classe dans la fenêtre qui s'affiche.
   La nouvelle classe s'affiche dans le diagramme.
2. Ouvrez la fenêtre de propriétés de la classe (menu contextuel de la classe/**Propriétés**).
3. Dans l'onglet **Caractéristiques**, champ **Stéréotype**, sélectionnez "XML Document Definition Root" à l'aide de la flèche.

Cette classe représente le schéma.

☺ *Chaque schéma XML est représenté dans **HOPEX** par une classe de stéréotype XML Document Definition Root. Lorsque vous créez un nouveau schéma XML à partir de l'éditeur de schémas XML (et non dans le diagramme), **HOPEX** crée automatiquement une classe avec le stéréotype XML Document Definition Root représentant le schéma.*

☺ *Si la classe n'a pas été créée dans un paquetage XML Document Definition, le paquetage détenteur de la classe est automatiquement modifié pour être un paquetage de stéréotype XML Document Definition.*

Vous pouvez aussi sélectionner **Nouveau > Schéma** dans le menu contextuel d'un paquetage. Dans ce cas, le stéréotype de la classe est automatiquement attribué et celui du paquetage également (si le paquetage ancêtre n'est pas déjà de stéréotype XML Document Definition).

Si vous créez la classe dans le diagramme et positionnez le stéréotype à la main, le paquetage détenteur n'est pas modifié pour avoir le stéréotype "XML Document Definition". Il est donc conseillé de suivre l'une des deux procédures citées plus haut.

DÉFINIR L'ALIAS D'UN PAQUETAGE

> *Un alias est un nom court donné à une urn par l'utilisateur (l'identifiant d'un espace de nommage).*

Les *alias* sont utilisés pour préfixer le nom des éléments d'un espace de nommage, dans les cas où ceux-ci sont utilisés dans un autre espace de nommage que le leur. Ils permettent d'identifier l'espace de nommage auquel ils appartiennent lorsqu'ils sont utilisés dans le cadre d'un espace de nommage différent.

Ainsi, si vous reliez un type appartenant à un autre espace de nommage à partir de votre schéma, le nom du type portera l'alias que vous avez défini suivi du nom du type.

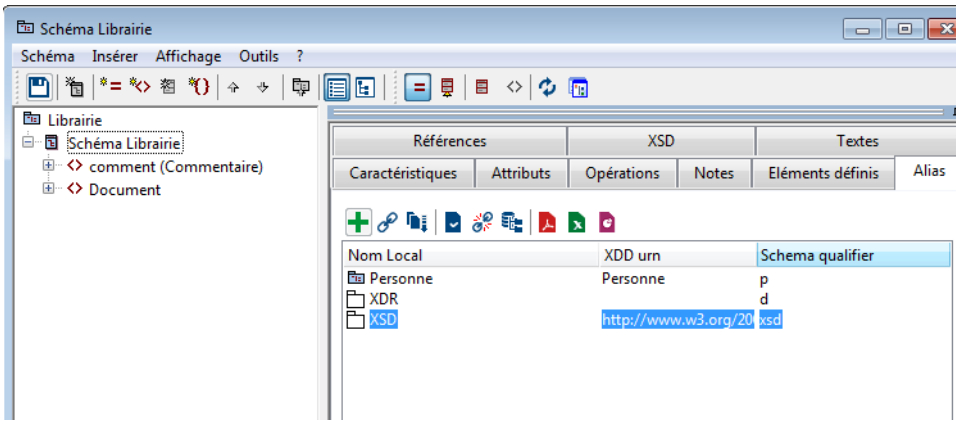
```
Exemple : d:string

Le schéma référence le type "string" représenté par l'alias
"d".
```

Cet alias est défini dans le cadre du schéma, et n'appartient pas à l'espace de nommage dans l'absolu : un autre utilisateur peut définir l'alias du paquetage comme il le souhaite.

Pour définir l'alias d'un paquetage :

1. Cliquez sur le schéma auquel vous voulez relier le type d'un autre paquetage.
2. Dans la fenêtre de propriétés, sélectionnez l'onglet **Alias**.
3. Cliquez sur le paquetage auquel appartient le type.
4. Dans le champ **urn**, entrez l'urn.
5. Dans le champ **Schema Qualifier**, définissez l'alias.



Pour relier une balise provenant d'un autre espace de nommage :

1. Cliquez sur le bouton **Relier** 
2. Dans la fenêtre de sélection, sélectionnez **Espaces de nommage ayant une urn**.

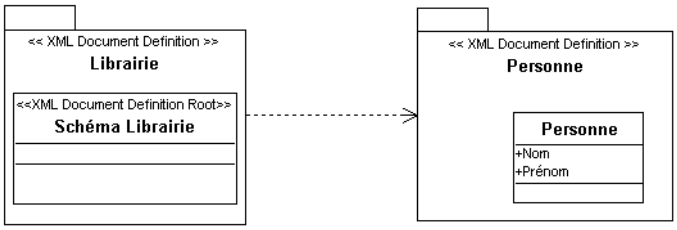
3. Dans la fenêtre qui apparaît, sélectionnez l'espace de nommage contenant les types, éléments et attributs à inclure dans le schéma.
4. Cliquez sur **Relier**.

Exemple

Le schéma "Librairie" veut utiliser la définition de balise "Personne". Pour ce faire, il se relie à l'espace de nommage "Personne", qui contient la définition de balise "Personne".

L'alias donné au nom de l'espace de nommage "Personne" ("Personne" étant l'urn de l'espace de nommage) auquel appartient "Personne", est p.

Le type "Personne" sera préfixé de p: dans le schéma "Librairie".



Dans le document, le résultat est :

```
<Librairie xmlns:p="Personne">
<p:Personne/>
</Librairie>
```

IMPORT ET INCLUSION D'AUTRES SCHÉMAS

Comme dans tout langage de programmation, il est important de pouvoir programmer de manière assez modulaire. Ainsi, on classe les éléments programmés par concept. Plusieurs concepts peuvent être assemblés dans une même bibliothèque.

Des bibliothèques standard sont ainsi fournies en XML. Dans chacune de ces bibliothèques se trouvent des centaines de schémas qui traitent chacun d'un concept particulier.

Les imports et inclusions de ces schémas sont traités différemment en XSD.

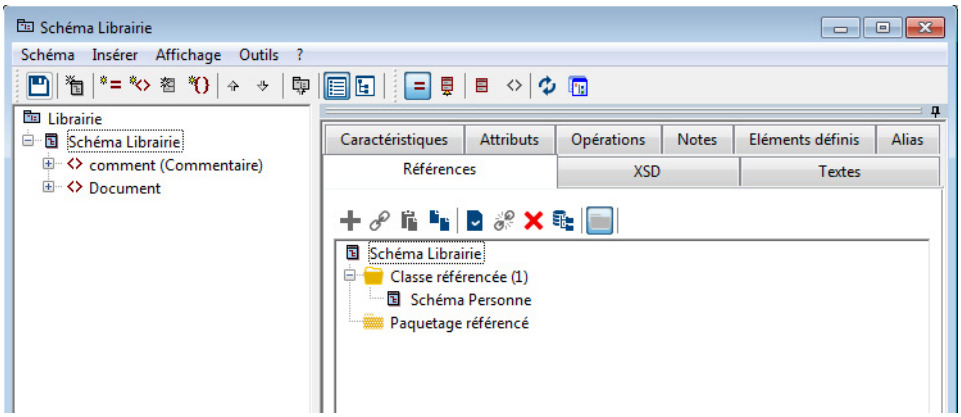
XSD

En XSD, vous pouvez référencer des types appartenant à d'autres espaces de nommage. Pour cela,vous devez référencer les espaces de nommage à l'aide de l'attribut xmlns. Cet attribut définit un alias sur un espace de nommage.

Cependant, cette référence ne suffit pas pour utiliser les types. Il faut également les inclure ou les importer dans le schéma courant. On distingue :

- L'inclusion : il s'agit d'inclure des schémas qui appartiennent au même espace de nommage que le schéma courant.
- L'import : il s'agit d'importer des schémas n'appartenant pas à l'espace de nommage du schéma courant ou des espaces de nommage entiers.

Dans l'exemple suivant, le schéma "Librairie" importe le schéma "Personne" qui appartient à un autre espace de nommage.



Pour importer le schéma d'un autre espace de nommage :

1. Sélectionnez le schéma dans lequel vous voulez importer un autre schéma.
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Références**.
3. Sélectionnez "classe référencée" ou "Paquetage référencé" selon que vous voulez importer un schéma particulier ou tous les schémas contenus dans un autre espace de nommage.

4. Cliquez sur le bouton **Relier**.
5. Recherchez le schéma ou le paquetage et cliquez sur **OK**.

Modélisation UML

Pour référencer un paquetage à partir du diagramme de classes :

1. Cliquez sur le menu **Affichage** > **Vues et détails** et cochez la case **Schéma**.
2. Cliquez sur le bouton **Lien**  de la barre d'objets.
3. Dans le diagramme, cliquez sur la classe schéma et reliez-la au paquetage à référencer.
4. Une fenêtre vous demande le type de lien que vous voulez créer. Sélectionnez **Paquetage référençé** et cliquez sur **OK**.

> ☛ *Vous pouvez de la même façon relier une classe à la classe schéma. Il s'agit d'un lien de type "Classe référencée".*

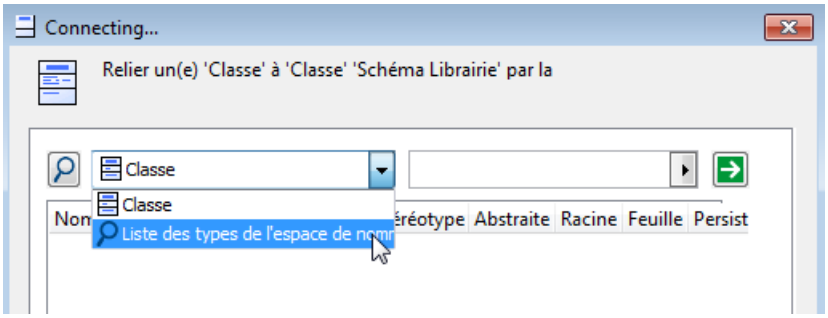
RELIER LA DÉFINITION DE BALISE À UNE CLASSE SCHÉMA

Lorsqu'un même espace de nommage comporte plusieurs schémas, il est nécessaire de relier les définitions de balises à la classe schéma à laquelle elles appartiennent.

> ☛ *Cette opération est facultative si l'espace de nommage n'est composé que d'un seul schéma. D'autre part, les éléments créés depuis l'éditeur de schémas sont automatiquement reliés au schéma.*

Pour relier une définition de balise :

1. Cliquez sur le schéma auquel vous voulez relier les balises. La fenêtre de propriétés de la classe apparaît dans la partie droite de l'éditeur.
2. Dans l'onglet **Eléments Définis**, cliquez sur le bouton **Relier** 
   La fenêtre de sélection apparaît.
3. Sélectionnez **Liste des types de l'espace de nommage**.



4. Dans la fenêtre qui apparaît, sélectionnez les types à relier au schéma et cliquez sur **Relier**.

Modélisation UML

- Ouvrez la fenêtre de propriétés de la classe **Schéma** à laquelle vous voulez relier des éléments (menu contextuel/**Propriétés**).
- Procédez tel que décrit ci-dessus.

DÉFINIR LES ÉLÉMENTS DANS UN SCHÉMA XML

Un élément représente une balise dans l'éditeur de schémas. Les éléments XML d'une balise A sont définis par l'ensemble des balises potentiellement incluses dans A.

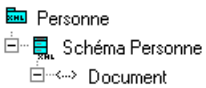
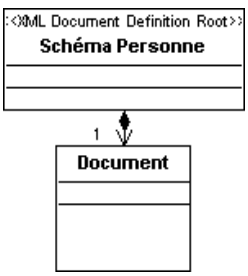
Créer un nouvel élément

Pour créer un nouvel *élément* :

1. Cliquez sur le schéma dans lequel l'élément doit être créé, puis cliquez sur le bouton **Création d'un élément** dans la barre d'outils.
   La fenêtre d'**Ajout d'un élément** s'affiche.

   ☛ *Le contenu de cette fenêtre varie suivant le mode sélectionné. Pour obtenir l'ensemble des propriétés d'un élément, vous devez sélectionner le "Niveau expert XML", accessible depuis le menu **Outils** de l'éditeur de schémas.*
2. Saisissez le nom de l'élément, renseignez les caractéristiques et cliquez sur **Créer**.

Résultat

Schéma	Diagramme	Document XML
		<Document>

Caractéristiques d'un élément

Lors de la création d'un élément, plusieurs champs sont à renseigner.

Multiplicité

La multiplicité d'un élément détermine le nombre d'apparitions de la balise incluse. Toutes les valeurs de multiplicité usuelles peuvent être utilisées, dont les valeurs courantes suivantes :

0: l'élément ne doit pas apparaître (utile pour restreindre une définition héritée),

1: l'élément doit apparaître obligatoirement une seule fois,

***:** l'élément est optionnel et peut être répété plusieurs fois,

1...*: l'élément est obligatoire et peut être répété plusieurs fois.

Pour préciser la multiplicité :

- Sélectionnez la multiplicité en vous aidant de la flèche.

Type d'élément

Dans ce champ, vous pouvez définir le type de l'élément (ou définition de l'élément). Par défaut, le nom du type est le même que celui de l'élément.

Type de base (facultatif)

Il s'agit du type dont le type de l'élément hérite.

Héritage (facultatif)

Correspond au type de l'héritage : restriction ou extension.

Local

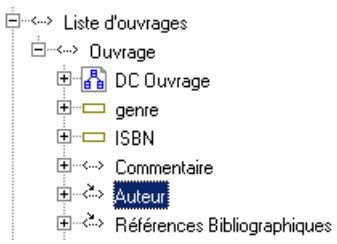
Local détermine si la définition de balise est créée localement, c'est-à-dire au niveau de la définition de balise contenant l'élément.

Si vous ne cochez pas cette case, la définition de balise est créée globalement, autrement dit au niveau de l'espace de nommage. Elle est alors réutilisable.

Accès par référence

Si vous cochez cette case, l'élément devient une référence. Voir "Références", page 48.

L'élément est créé sous le schéma dans le navigateur.



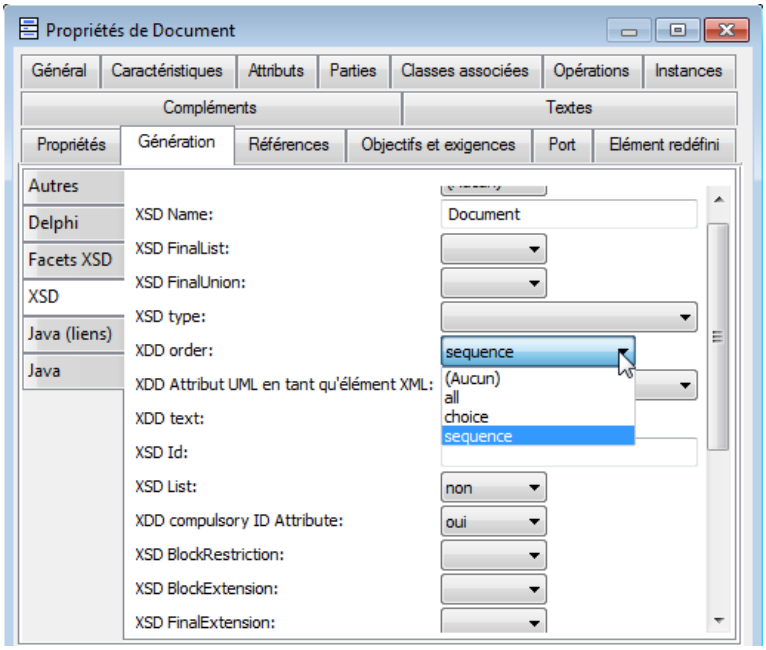
Ordonner les éléments contenus

Pour spécifier dans quel ordre sont générés les éléments contenus :

1. Faites un clic droit sur l'élément à ordonner, et choisissez **Propriétés du type** dans le menu contextuel.

   ☛ *Vous devez être en mode "Niveau expert XML" pour voir apparaître cette commande.*

   La fenêtre de propriétés s'affiche.
2. Dans l'onglet **Génération**, dans le sous-onglet **XSD**, renseignez le champ **XDD Order** à l'aide de la flèche :
   - "All" : les éléments du type apparaissent aucune ou une fois et dans n'importe quel ordre. Les groupes "all" ne sont autorisés qu'au plus haut niveau d'un type et doivent être seuls. De plus, ils ne peuvent contenir que des éléments.
   - "Sequence" : les balises doivent apparaître dans l'ordre, et le nombre de fois spécifié par la multiplicité de l'élément.
   - "Choice" : une seule des sous-balises peut apparaître. Le nombre de fois est spécifié par la multiplicité.



Modélisation UML

- Affichez la fenêtre de propriétés de la classe (menu contextuel/ **Propriétés**) dont vous voulez ordonner les classes composantes et procédez tel que décrit ci-dessus.

DÉFINIR LES ATTRIBUTS

Un attribut XML d'une balise est une paire de chaînes de caractères nom-valeur. La deuxième chaîne est entourée par des guillemets. Les attributs XML servent à caractériser la balise. Exemple : <personne nom = 'Dupont'/>

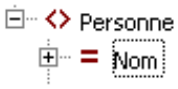
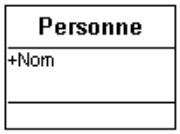
Un attribut ne contient pas d'autres informations que celles liées à sa valeur.

Créer un nouvel attribut dans un schéma XML

Pour créer un nouvel *attribut* :

1. Sélectionnez l'élément auquel vous voulez donner un nouvel attribut, puis cliquez sur le bouton **Création d'un Attribut** dans la barre d'outils de l'éditeur.
   La fenêtre de création d'un attribut s'affiche.
2. Saisissez le nom de l'attribut et cliquez sur **OK**.
   L'attribut est créé.

Résultat

Schéma	Diagramme	Document XML
		<personne nom = "Dupont"/>

Modélisation UML

Pour ajouter des attributs :

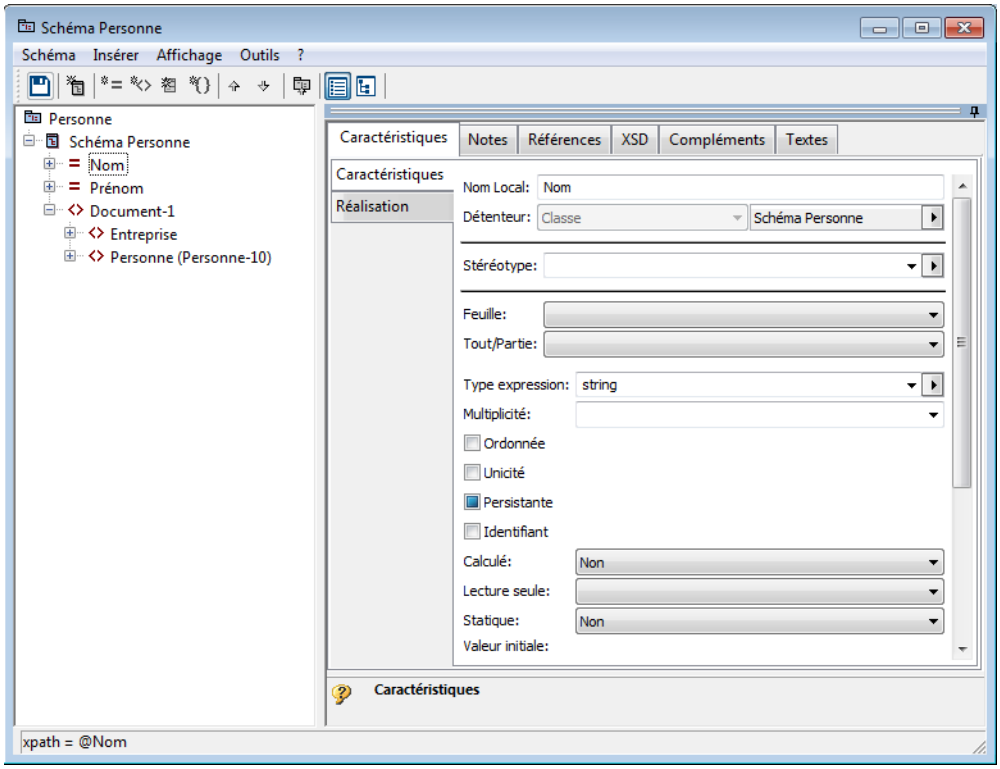
1. Ouvrez la fenêtre de propriétés de la classe pour laquelle vous voulez créer un attribut (menu contextuel/ **Propriétés**)
2. Dans l'onglet **Attribut**, cliquez sur le bouton **Nouveau** .
3. Saisissez le nom de l'attribut et validez.

Caractéristiques d'un attribut

Pour définir les caractéristiques de l'attribut :

- Sélectionnez l'attribut dans la partie gauche de l'éditeur de schéma. Dans la partie droite, la fenêtre des propriétés de l'attribut s'affiche.

  ☛ *Vous pouvez également accéder à sa fenêtre de propriétés en cliquant avec le bouton droit sur l'attribut et en sélectionnant* ***Propriétés****.*



Type de l'attribut

Le champ **Type expression** vous permet de définir le type de l'attribut. La liste qui s'affiche dans ce champ correspond aux types du paquetage courant et aux types de l'espace de nommage référencé.

Valeur initiale

Chaque attribut peut disposer d'une valeur par défaut. Lorsqu'une balise omet de définir un attribut optionnel, et si ce dernier dispose d'une valeur par défaut, cette valeur lui sera attribuée automatiquement.

Pour définir une valeur par défaut :

- Ouvrez la fenêtre de propriétés de l'attribut.
- Cliquez sur l'onglet **Caractéristiques**.
- Dans le champ **Valeur initiale**, saisissez la valeur par défaut.

Exemple

La classe Collection possède un attribut Nom, dont la valeur initiale est Portfolio par défaut.

XSD

```
<xsd:ComplexType name="Collection">
    < xsd:attribute name="nom" default="PortFolio"
use="required">
      < xsd:SimpleType>
      <xsd :restriction base= 'xsd :string'>
        <xsd:enumeration value='PortFolio'/>
        <xsd:enumeration value='Poche'/>
        <xsd:enumeration value=' Larousse'/>
      </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
  </xsd :ComplexType>
```

Multiplicité

> 🕮 *La multiplicité d'un attribut définit le nombre d'occurrences de l'attribut dans une classe.*

Dans le cadre d'un schéma, un attribut ne peut apparaître plus d'une fois (une **Personne** n'a qu'un seul nom par exemple). Les multiplicités qu'il est possible de spécifier sont donc :

0 : attribut interdit

0..1 : optionnel

1 : obligatoire (valeur par défaut)

- Dans le champ **Multiplicité**, saisissez **0**, **0**…**1 ou 1**.

> ☛ *La Multiplicité 0 est utile lorsqu'une classe hérite d'un attribut non désiré : grâce à la fonction de surcharge , il est possible de positionner la multiplicité de l'attribut sur 0 et donc d'écarter l'attribut. Voir le chapitre "Compléments liés à l'héritage", page 70.*

Statique

Il est possible de spécifier si l'attribut peut être modifié. Le champ **Statique** indique si attribut peut prendre des valeurs spécifiques pour chacune des instances de la classe ou bien avoir une valeur qui caractérise l'ensemble de la classe.

- "Oui" : l'attribut a une valeur qui caractérise l'ensemble de la classe. Par exemple, l'attribut "Longueur des numéros de téléphone" de la classe "Client France" est de 10 chiffres.
- "Non" : l'attribut peut prendre une valeur différente pour chacune des instances de la classe. Par exemple, l'attribut "Numéro de téléphone" prend une valeur différente pour chaque instance de la classe "Client".

Exemple

Dans le cas de l'attribut **Collection**, si le champ est positionné sur :

- "Oui" : la valeur de l'attribut **Collection** ne peut pas être modifiée.
- "Non" : la valeur "Portfolio" peut être changée en "Larousse" par exemple.
- "Ajout seul" : ce paramètre n'est pas utilisé dans le cadre des schémas XML. Si vous le sélectionnez, l'attribut est considéré comme modifiable.

Modélisation UML

Pour définir les propriétés d'un attribut :

1. Ouvrez la fenêtre de propriétés de la classe (menu contextuel > **Propriétés**) et allez dans l'onglet **Attribut**.
2. Sélectionnez l'attribut concerné et cliquez sur le bouton **Propriétés**  . La fenêtre de propriétés de l'attribut s'affiche.

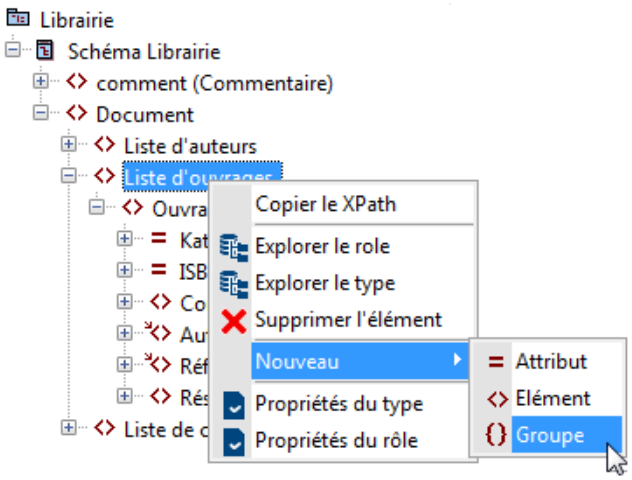
GROUPES

Un groupe est un ensemble d'éléments ou d'attributs. Il est utile de constituer des groupes pour pouvoir ordonner les éléments qu'ils contiennent.

Créer un groupe dans un schéma XML

Pour créer un nouveau *groupe* :

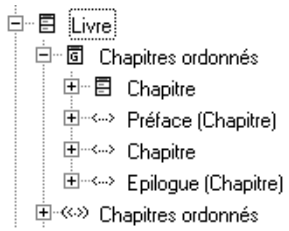
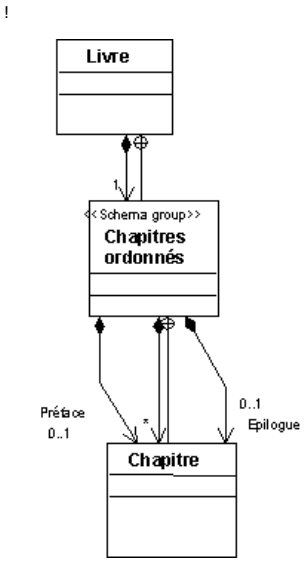
1. Dans le menu contextuel de l'élément dont vous voulez ordonner les balises dans le navigateur XML, sélectionnez **Nouveau** > **Groupe**.



2. Dans la fenêtre qui apparaît, saisissez le nom du groupe.
3. Cliquez sur **Créer**.

> ☛ *La fenêtre de création d'un groupe varie suivant le mode sélectionné. Pour obtenir l'ensemble des propriétés d'un groupe, vous devez sélectionner le "Niveau expert XML", accessible depuis le menu **Outils** de l'éditeur de schémas.*

Exemple

Schéma	Diagramme	Document XML
		<Livre> <Preface/> <Chapitre> </Chapitre> ... <Epilogue> </Epilogue> </Livre>

Modélisation UML

Un groupe est représenté par une classe de stéréotype **Schema Group**.

- Créez une classe.
- Ouvrez sa fenêtre de propriétés.
- Cliquez sur l'onglet **Caractéristiques**.
- Dans le champ **stéréotype**, sélectionnez "Schema Group" .

Groupe d'attributs modèle

Il est possible de définir des modèles de groupes d'attributs. Ce type de groupe ne contient pas d'élément et contient un certain nombre d'attributs et de sous-groupes d'attributs. Il n'existe pas de notion d'ordre pour les groupes d'attributs.

Ces groupes apparaissent au niveau des schémas et sont ensuite réutilisés au sein des définitions de balise par référencement. Cela évite la redondance de définition.

Groupe d'éléments modèle

Comme pour les attributs, il est possible de définir des modèles de groupes d'éléments. Le but est d'éviter la redondance de définition car l'héritage entre groupes n'existe pas. Ces groupes sont définis au niveau du schéma. Il contiennent

des sous-groupes d'éléments (All, sequence, choice). Ils sont référencés au sein des définitions de balises.

Ordonner les éléments contenus dans un groupe

Les éléments d'un groupe peuvent être ordonnés.

Pour ordonner les éléments :

- Sélectionnez le groupe dans le navigateur de l'éditeur de schémas.
- Dans la partie droite, cliquez sur l'onglet **Caractéristiques**.
- Dans le champ **Ordonnancement**, sélectionnez l'une des valeurs suivantes à l'aide de la flèche :
  - "All" : les éléments du type apparaissent aucune ou une fois et dans n'importe quel ordre. Les groupes "all" ne sont autorisés qu'au plus haut niveau d'un type et doivent être seuls. De plus, ils ne peuvent contenir que des éléments.
  - "Sequence" : les éléments doivent apparaître dans l'ordre spécifié par l'attribut de lien Order entre la classe **Schema Group** et les rôles d'association.
  - "Choice" : seul un des éléments doit apparaître.

Vous pouvez également ordonner les éléments dans la fenêtre de création du groupe. Voir "Créer un groupe dans un schéma XML", page 43.

Ou :

- Dans l'éditeur de schémas, cliquez avec le bouton droit sur le groupe à ordonner et sélectionnez **Propriétés du type**.
  La fenêtre de propriétés s'affiche.
- Dans l'onglet **Génération**, dans le sous-onglet du langage de génération (**XSD**) renseignez le champ **XDD Order** à l'aide de la flèche.

La valeur du paramètre **XDD Order** et celle des multiplicités des sous-éléments définissent la nature de l'ordonnancement et la fréquence d'apparition des sous-éléments au sein du groupe.

Le tableau suivant présente quelques exemples d'utilisation :

XDD Order	Multiplicité de A	Multiplicité de B	Multiplicité de C	Commentaire	Exemple
Choice	1	*	1	Une balise A ou (0 ou plusieurs B) ou une C	<X> <A/> </X>
Sequence	1	1	*	Une balise A suivie d'une B suivie de 0 ou plusieurs C.	<X> <A/> <B/> <C/> <C/> </X>
All	1	1	1	Une balise de chaque dans n'importe quel ordre.	<X> <B/> <C/> <A/> </X>

Modélisation UML

- Affichez la fenêtre de propriétés d'une classe de stéréotype **Schema Group** et procédez tel que décrit ci-dessus.

Utiliser un groupe

Les groupes peuvent être utilisés comme les classes pour déclarer un ensemble.

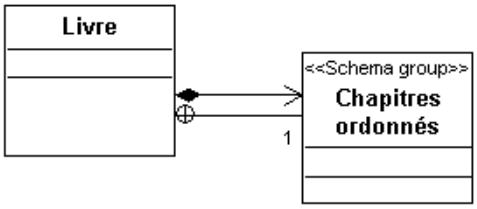
En XSD, une balise spéciale est créée (<sequence>, <choice>, <all>).

Exemple

Les balises définies sur la classe "Livre" déclarent une instance de groupe (multiplicité égale à 1). D'autre part, afin que le paquetage schéma ne soit pas

"surpeuplé" inutilement, la classe "Chapitres ordonnés" est déclarée localement à la classe "Livre".

> *Local : cette caractéristique détermine si le groupe est créé localement à la définition de balise ou au groupe contenant l'élément, ou s'il est créé globalement (au niveau de l'espace de nommage).*



XSD

```
  <xsd:ComplexType name="Livre" >
     <xsd:sequence>
    <xsd:element name="Resume" type='Resume' minOccurs="1"
maxOccurs="1"/>
    <xsd:sequence minOccurs="1" maxOccurs="1">
      <xsd:element name = 'Preface' type="Preface"
minOccurs="0" maxOccurs="1"/>
      <xsd:element name = 'Chapitre' type="Chapitre"
minOccurs="0" maxOccurs="*"/>
      <xsd:element name = 'Epilogue' type="Epilogue"
minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
   </xsd:sequence>
   <xsd:attribute name="ISBN" type="xsd:ID" use="required"/
>
   <xsd:attribute name="Auteur" type="xsd:IDREF"
use="required"/>
   <xsd:attribute name="Collection" type="xsd:IDREFS" />
   <xsd:attribute name="genre" type="genre" use="required"/
>
  </xsd:ComplexType>
```

RÉFÉRENCES

Une référence permet de créer une relation entre deux balises sans nécessiter l'inclusion de la balise référencée : si un document contient une liste de personnes, on pourra faire référence aux personnes déclarées dans cette liste, sans avoir à émettre leur déclaration. Ce principe évite la redondance d'information et facilite la maintenance du document.

Créer une référence dans un schéma XML

Dans l'éditeur de schéma, la référence ne peut être créée que depuis un élément.

Pour créer une *référence* :

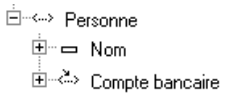
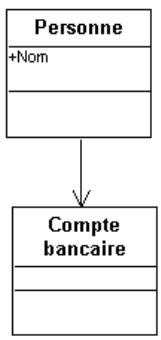
- Cochez la case **Accès par Référence** dans la fenêtre de création d'un élément.

☛ *Voir "Créer un nouvel élément", page 36.*

ou :

- Cliquez sur l'élément que vous voulez définir comme étant une référence dans le navigateur.
- Dans la partie droite de l'éditeur, sous l'onglet **Caractéristiques**, cochez la case **Accès par référence**.

Résultat

Schéma	Diagramme	Document XML
		<Personne Compte="1306" Nom="Dupont"/> <CompteBancaire id="1306"/>

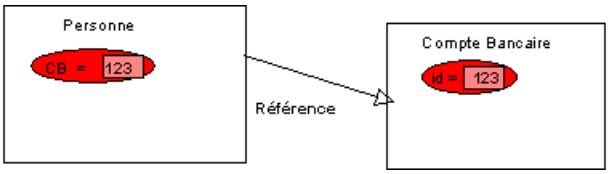
Principe de fonctionnement

L'élément défini comme une référence porte automatiquement un attribut qui permet de l'identifier pour s'y référer : l'attribut id, de type ID.

> ☛ *Ce type est défini par XML ; il est donc disponible dans tous les langages de schémas.*

La définition de balise faisant référence à cet élément va automatiquement définir un attribut de type **IDREF** ou **IDREFS** (selon la multiplicité), dont la valeur sera la valeur de l'attribut **id** de l'élément référencé.

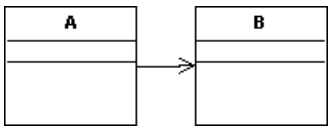
L'attribut **id** est ajouté par défaut à toutes les instances de balises de la définition de balise.



> ☛ *L'ajout de cet attribut d'identification est conditionné par la valeur du paramètre **XDD compulsory ID attribute**. La valeur par défaut (**oui**) indique que l'attribut doit être ajouté.*

Exemple

Lorsqu'une classe A fait référence à une classe B, un attribut de type **IDREF** (ou **IDREFS** selon la multiplicité de la référence) est ajouté à la classe A. La valeur de cet attribut doit correspondre à la valeur de l'attribut **ID** d'une balise B.

Schéma	Document XML
	<A id="25", B="123"/> <B id="123"/>

Modélisation UML

La référence est représentée par une association comparable à celle utilisée pour la déclaration d'élément. La seule différence est dans la nature de l'association : ce n'est pas une composition.

- Créez une association entre la classe référencée et la classe qui référence.
- Faites un clic droit sur le rôle de l'association qui référence et sélectionnez **Agrégation** > **Non** (valeur cochée par défaut).

Caractéristiques d'une référence

Pour définir certaines caractéristiques de l'attribut ID ajouté à l'élément référencé :

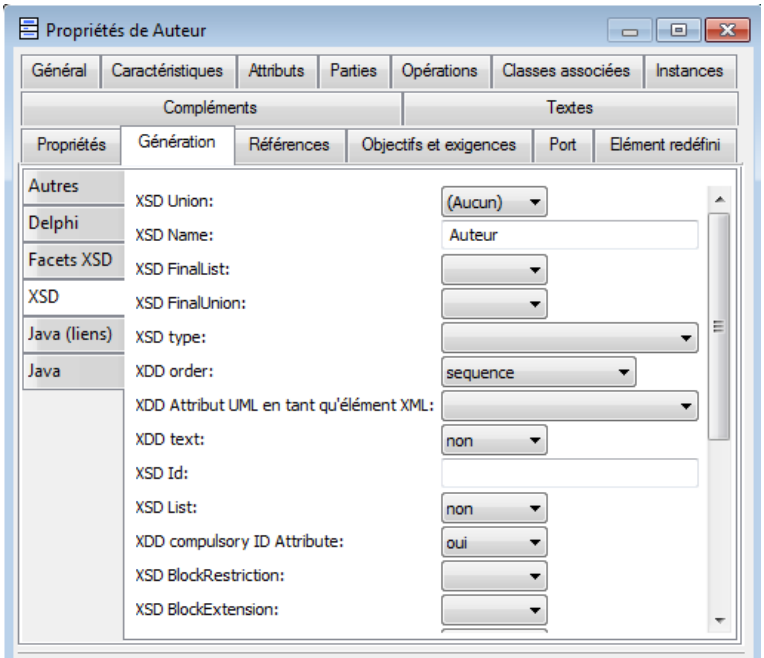
1. Faites un clic droit sur l'élément référencé et sélectionnez **Propriétés du type**.

   ☛ *Les "Propriétés du type" sont les propriétés de la classe qui définit l'élément.*

   Les "Propriétés du rôle" sont les propriétés du rôle de l'association correspondant à l'élément.

   Voir "Elément", page 16.

2. Cliquez dans l'onglet **Génération** et allez dans le sous-onglet du langage de génération que vous utilisez.



3. Sélectionnez la valeur des paramètres ci-dessous en cliquant dans leur champ respectif et en vous aidant de la flèche.
   - Paramètre **XDD compulsory ID attribute** : détermine si l'attribut est à ajouter à la définition de balise.

     ☛ *En XSD, un attribut de type ID créé sur le type est prioritaire par rapport à un attribut ID défini dans les paramètres de génération. L'utilisation de ce type de référence est déconseillée en XSD.*

Modélisation UML

La valeur de cet attribut peut être défini globalement sur un des paquetages pères de la classe. Elle est alors répercutée sur toutes les classes détenues par le

paquetage et les sous-paquetages. Cette valeur peut alors être redéfinie localement.

- **Paramètre XDD ID Attribute name** : définit le nom de l'attribut (à saisir manuellement).
- **Paramètre XDD ID Attribute required** : précise si la valeur de l'attribut doit obligatoirement être définie dans le document. "Non" est sélectionné par défaut.

Créer une référence bidirectionnelle

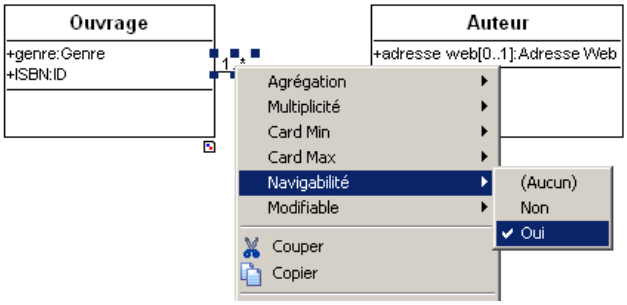
Il est possible de référencer deux définitions de balises par la même association.

Il faut alors créer une association bidirectionnelle (navigable dans les deux sens) entre les deux classes représentant le schéma et l'élément référencé.

Vous devez également renseigner les multiplicités des deux rôles (par défaut, la valeur de la multiplicité est *).

Pour créer une référence bidirectionnelle :

1. Ouvrez un diagramme, en faisant un clic droit sur le schéma et en sélectionnant **Nouveau > Diagramme de classes**.
2. Glissez les deux éléments de l'éditeur de schémas dans la fenêtre du diagramme.
3. Dans le menu contextuel du rôle de l'association non navigable, sélectionnez **Navigabilité** > **Oui**.



La flèche disparaît du lien : la référence est devenue bidirectionnelle.

4. Fermez le diagramme en enregistrant.

Fonctionnement

Dans le cas d'une association bidirectionnelle, un attribut de type **IDREF** (ou **IDREFS**) est généré pour chaque balise basée sur les deux types reliés. Dans le document instance, la valeur de cet attribut fait référence à un attribut de type **ID** de la balise opposée.

Exemple :

Dans l'exemple ci-dessous, chaque ouvrage référence les auteurs et inversement.

Dans les sources générées, seuls les attributs permettant de faire la référence des ouvrages depuis un auteur sont en gras. Les attributs assurant la référence symétrique sont facilement repérables.

XSD

```
<xsd:complexType name="Ouvrage" abstract="false"
mixed="false">
    <xsd:sequence>
      <xsd:element name="Resume" abstract="false"
minOccurs="1" maxOccurs="1" type="Resume">
      </xsd:element>
      <xsd:element ref="comment"/>
    </xsd:sequence>
    <xsd:attribute name="genre" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:anyType">
          <xsd:enumeration value="Pratique"/>
          <xsd:enumeration value="Science Fiction"/>
          <xsd:enumeration value="Littérature"/>
          <xsd:enumeration value="Musique"/>
          <xsd:enumeration value="Scolaire"/>
          <xsd:enumeration value="Enfance"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="ISBN" type="xsd:ID"
use="required">
    </xsd:attribute>
    <xsd:attribute name="ReferencesBibliographiques"
use="required" type="IDREF"/>
    <xsd:attribute name="Auteur" use="required"
type="IDREF"/>
  </xsd:complexType>
```

Référence à un élément

Il est possible de définir un élément au niveau du schéma. Cet élément peut ensuite être référencé au sein d'une définition de balise. Le but est d'éviter la redondance de définition. Ainsi, on peut par exemple définir un élément "comment" au niveau du schéma et réutiliser cet élément dans les différentes définitions de balises du schéma.

Pour référencer un élément dans l'éditeur de schémas :

1. Sélectionnez l'élément dans le navigateur.
2. Dans la partie droite de l'éditeur, cliquez sur l'onglet **Références**.
3. Sélectionnez **Elément référencé** et cliquez sur le bouton **Relier**.

4. Recherchez l'élément à référencer et cliquez sur **OK**.

Modélisation UML

Pour référencer un élément à partir du diagramme de classes :

1. Cliquez sur le menu **Affichage** > **Vues et détails** et cochez la case **Schéma**.
2. Cliquez sur le bouton **Lien**  de la barre d'objets.
3. Dans le diagramme, cliquez sur le rôle référençant puis reliez-le au rôle référencé.
4. Une fenêtre vous demande le type de lien que vous voulez créer. Sélectionnez **Elément référencé** puis cliquez sur **OK**.

Référence à un attribut

Il est possible de définir un attribut au niveau du schéma. Cet attribut peut ensuite être référencé au sein d'une définition de balise. Le but est d'éviter la redondance de définition.

Pour référencer un attribut dans l'éditeur de schémas :

1. Sélectionnez l'attribut dans le navigateur.
2. Dans la partie droite de l'éditeur, cliquez sur l'onglet **Références**.
3. Sélectionnez **Attribut référencé** et cliquez sur le bouton **Relier**.
4. Dans la fenêtre qui apparaît, recherchez l'attribut à référencer et cliquez sur **OK**.

Référence à un groupe

Lorsqu'on définit un groupe modèle, il a pour but d'être référencé.

☛ *Voir "Groupe d'attributs modèle", page 44.*

Pour référencer un groupe dans l'éditeur de schéma :

1. Sélectionnez l'élément groupe dans la partie gauche de l'éditeur de schémas.
2. Dans la partie droite, cliquez sur l'onglet **Références**.
3. Sélectionnez **Elément référencé** et cliquez sur le bouton **Relier**.
4. Dans la fenêtre qui apparaît, recherchez le groupe à référencer et cliquez sur **OK**.

UTILISER LES PAQUETAGES POUR LE CLASSEMENT

Les paquetages n'ayant pas le stéréotype **XML Document Definition** peuvent être utilisés :

- soit pour classer des espaces de nommage
- soit pour partitionner un espace de nommage

L'utilisation des paquetages non **XML Document Definition** n'a aucun effet sur la sémantique des schémas générés.

Ainsi vous pouvez créer de nouveaux paquetages non-XML Document Definition dans votre schéma, et y insérer des éléments ou espaces de nommage, sans que ces paquetages soient générés.

Attention, la création de nouveaux paquetages peut être source d'erreur lors de la rétro-génération.

COMPLÉMENTS XSD

Le schéma XML, ou XSD (XML Schema Definition), est un standard défini par le W3C. Fondé sur XML, XSD permet de définir la structure de documents XML.

Le but d'un schéma est de définir une classe de documents XML, et donc le terme "document instance" est souvent utilisé pour définir un document qui est valide par rapport à un certain schéma.

Afin de permettre au schéma XML de faire référence aux types **XSD**, il est nécessaire d'importer la bibliothèque correspondante. Voir "Importer le Solution Pack XSD", page 14.

DÉFINITIONS DE BALISES

Définitions simples

Les définitions simples de balises correspondent à des balises typées sans attribut ni élément.

En XSD, l'utilisateur a la possibilité de définir des types simples qui restreignent les types standard. Ainsi, on distingue trois types de définition pour les types simples XSD.

Définition d'union

Un type simple peut être une union de types. Ceci peut être valable lorsque plusieurs pays n'utilisent pas le même format pour définir un concept particulier. Ainsi, si on prend l'exemple du code postal, il est défini de manière différente en France et aux Etats-Unis.

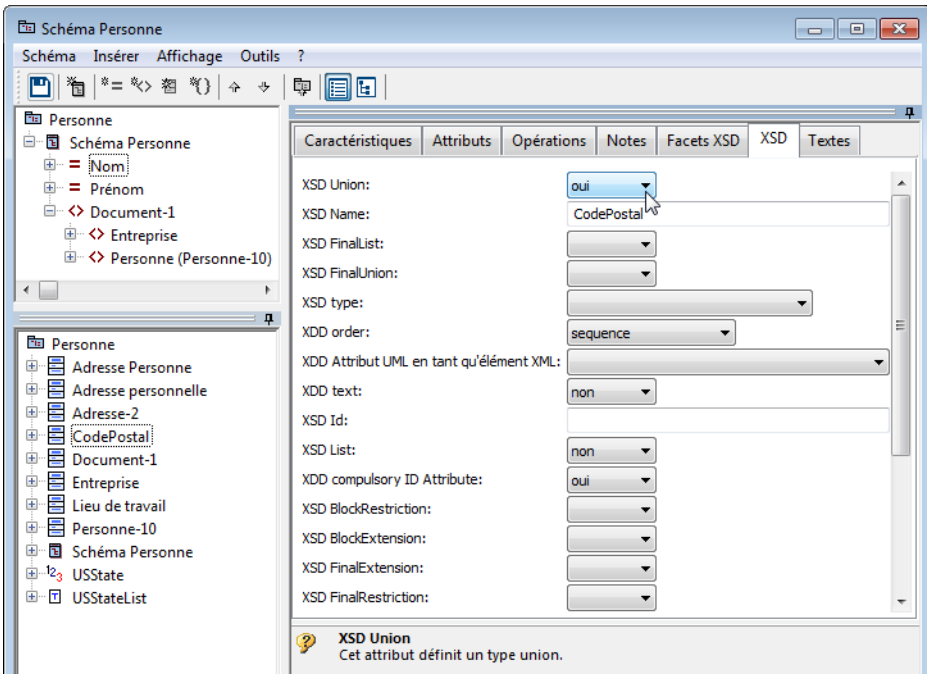
```
<xsd:simpleType name="USState">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AK"/>
    <xsd:enumeration value="AL"/>
    <xsd:enumeration value="AR"/>
    <!-- and so on ... -->
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="zipUnion">
  <xsd:union memberTypes="USState">
     <xsd:simpleType>
       <xsd:list itemType="xsd :int"/>
     </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>
```

Pour définir une union de types :

1. Dans l'éditeur de schéma, cliquez sur **Affichage** > **Type**.
2. Dans le navigateur, sélectionnez la classe (le type) pour laquelle vous voulez définir une union des types (ici, "code postal").
3. Dans la fenêtre de **Propriétés**, cliquez sur l'onglet **Caractéristiques**.
4. Dans le champ **Stéréotype**, sélectionnez "structure".
5. Cliquez sur l'onglet **XSD**.

6. Dans le champ **XSD Union**, sélectionnez "oui".

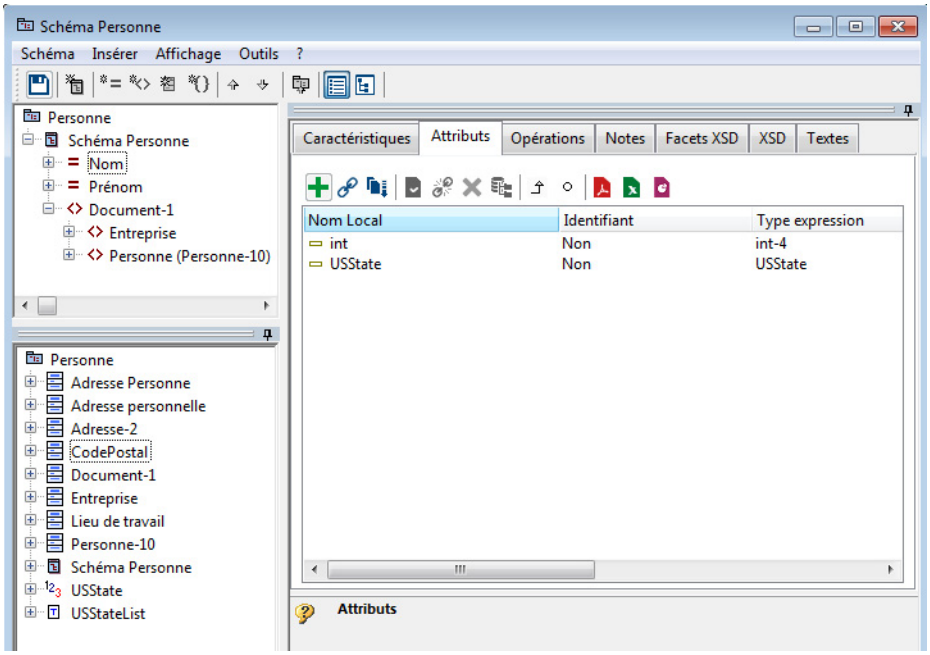


Les types définis dans l'union sont montrés en tant qu'attributs de la classe.

Pour définir les types qui interviennent dans l'union :

1. Sélectionnez dans le navigateur la classe représentant l'union.
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Attributs**.
3. Cliquez sur le bouton **Nouveau**.
4. Créez vos attributs. Donnez-leur de préférence le même nom que les types de l'union.

5. Dans la colonne **Type expression**, sélectionnez les types définissant l'union. Ici, il s'agit des types "int" et "USState".

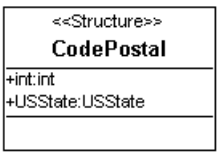


Modélisation UML

Pour créer une union de types dans un diagramme :

1. Cliquez avec le bouton droit sur la classe et sélectionnez **Propriétés**.
2. Sous l'onglet **Caractéristiques**, sélectionnez le **stéréotype** "Structure".
3. Cliquez sur l'onglet **Génération** et sur le sous-onglet **XSD**.
4. Dans le champ **XSD Union**, sélectionnez "oui".
5. Définissez ensuite de la même façon les types de l'union.

Dans notre exemple, la classe "CodePostal " est une union des types "int" et "USState". Ces types apparaissent sous forme d'attributs.



Définition de restriction (Facets)

Un type simple peut être une restriction d'un autre type simple ou d'un type standard XSD. Par restriction, on entend "Enumération", limitation de la longueur d'une chaîne de caractères, définition d'un intervalle de valeurs etc.

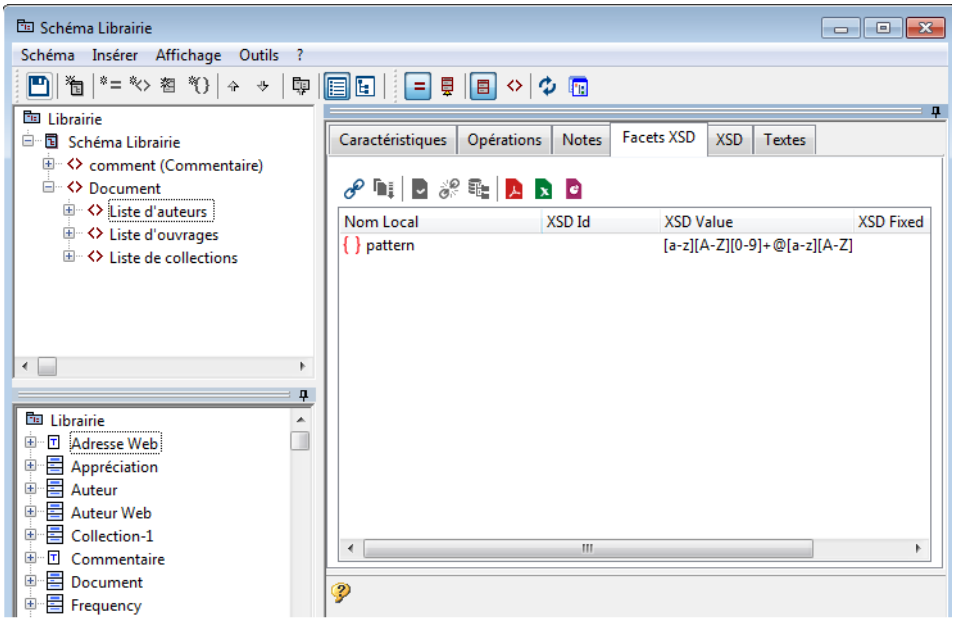
```
<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

Dans cet exemple, on définit un type sku. Il s'agit d'une string qui doit commencer par 3 chiffres suivi d'un tiret puis de 2 caractères ascii.

Les facets sont modélisées par des contraintes sur classe. Chaque facet correspond à une occurrence de contrainte.

Pour définir une facet sur un type :

1. Dans l'éditeur de schémas, sélectionnez la classe (le type).
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Facets XSD**.
3. Cliquez sur le bouton **Relier** 
4. Sélectionnez la liste des facets et cliquez sur **OK**.
5. Dans la liste qui apparaît, sélectionnez la facet que vous voulez relier à la classe et cliquez sur **OK**.
6. Dans la colonne "XSD Value", entrez la valeur de la facet.



Cas particulier de l'énumération

Pour modéliser une énumération :

1. Ouvrez la fenêtre de propriétés de la classe.

2. Dans l'onglet **Caractéristiques**, sélectionnez le **stéréotype** "Enumération".
3. Cliquez sur **OK**.
4. Ouvrez de nouveau la fenêtre de propriétés de la classe. Un nouvel onglet **Valeur littérale** est apparu.
5. Entrez sous cet onglet les valeurs de l'énumération.

Voir aussi "Stéréotype Enumération", page 79.

Définition d'un type "Liste"

Un type simple peut également être défini comme une liste d'un type simple. Cela permet de définir des listes ou des tableaux pouvant prendre comme valeur une liste d'éléments d'un type simple.

```
<xsd:simpleType name="USStateList">
    <xsd:list base="USState">
    </xsd:list>
  </xsd:simpleType>
```

Pour définir un type comme une liste de valeurs :

1. Dans le navigateur de l'éditeur, sélectionnez le type en question.
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Caractéristiques**.
3. Dans le champ **Stéréotype**, sélectionnez "Expression".
4. Cliquez sur l'onglet XSD.
5. Dans le champ **XSD List**, sélectionnez "Oui". Rafraîchissez la fenêtre de propriétés du type et retournez dans l'onglet **Caractéristiques**.
6. Dans le champ **Type expression**, sélectionnez le type de valeurs que vous voulez définir pour le type. Par exemple, si vous voulez que le type soit une liste d'entiers, sélectionnez "int".

Définitions complexes

Les définitions complexes de balises correspondent à des balises ayant des attributs et des éléments.

En XSD, on distingue également les types qui ne contiennent que des attributs (types à contenu simple) et ceux qui contiennent des attributs et des éléments (types à contenu complexe).

On peut appliquer sur ces types les règles d'héritage, de polymorphisme. Voir "Héritage", page 66.

LES CONTRAINTES

On peut définir des contraintes sur les éléments ou balises. Ces contraintes s'appliquent aux attributs ou au contenu de ces balises. Les contraintes définies sur les balises sont l'unicité et la mise en correspondance par le biais de clés primaires et de clés étrangères.

Unicité

L'unicité permet de caractériser de manière unique un élément. L'unicité est modélisée par une contrainte sur le rôle d'une classe. Elle a pour stéréotype "XSD Uniqueness".

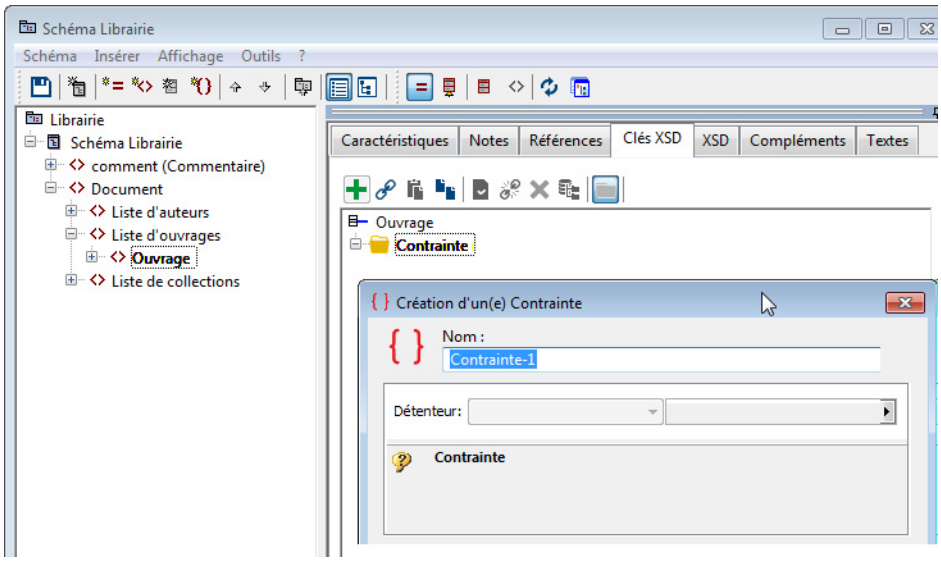
Pour définir l'unicité d'un élément :

Depuis l'éditeur de schéma :

- Sélectionnez l'élément dans le navigateur.
- Dans la fenêtre de propriétés, cliquez sur l'onglet **Clés XSD**.

Pour créer une contrainte :

- Sélectionnez "Contrainte".
- Cliquez sur le bouton **Nouveau**
- Dans la fenêtre qui apparaît, entrez le nom de la contrainte et cliquez sur **OK**.



Pour relier une contrainte existante :

- Sélectionnez "Contrainte".
- Cliquez sur le bouton **Relier**.
- Dans la fenêtre qui s'affiche, recherchez le nom de la contrainte et cliquez sur **Relier**.

La contrainte apparaît dans la fenêtre de propriétés.

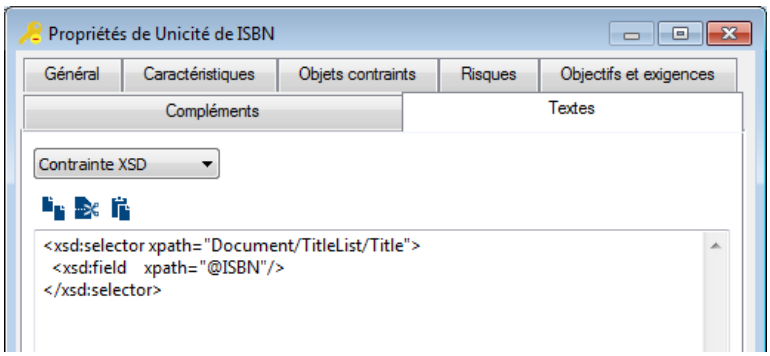
Pour attribuer le type "unicité" à la contrainte :

- Cliquez avec le bouton droit sur la contrainte.
- Cliquez sur **Propriétés**.
- Dans l'onglet **Caractéristiques**, sélectionnez le **Stéréotype** "XSD Uniqueness".

Précisez ensuite à quoi s'applique l'unicité :

- Dans la fenêtre de propriétés de la contrainte, cliquez sur l'onglet **Textes**.
- Sélectionnez "Contrainte XSD".
- Précisez les éléments ou attributs qui définissent l'unicité de l'élément (field) ainsi que le chemin de ces attributs et éléments (selector). Voir les balises "Field" et "selector" définis par les schémas du W3C. Les schémas XSD utilisent le xpath pour caractériser un attribut.

Dans notre exemple, la classe "Ouvrage" est définie de manière unique par l'attribut "ISBN".



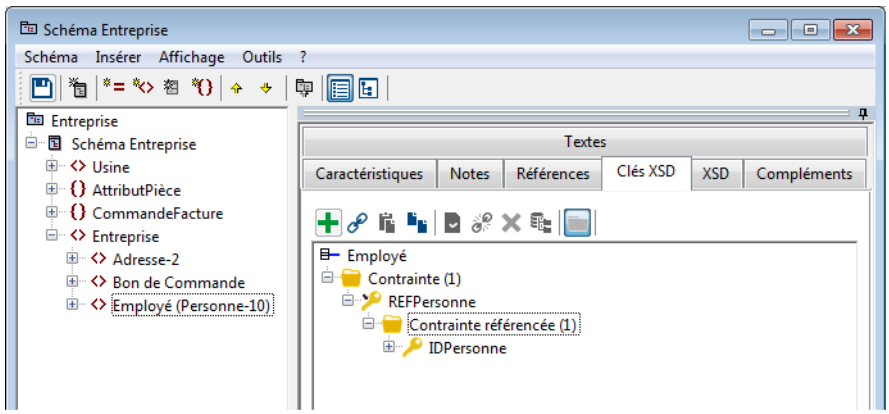
Clé primaire, Clé étrangère

Vous pouvez définir une contrainte d'unicité (clé primaire) et mettre en correspondance cette contrainte avec une ou plusieurs clés étrangères. Ainsi, les éléments possédant cette clé étrangère ne peuvent exister si leur clé ne prend pas ses valeurs dans celles proposées par la clé primaire.

La clé primaire est modélisée par une contrainte de stéréotype "XSD Key". La clé étrangère est modélisée par une contrainte de stéréotype "XSD Key Reference". Pour caractériser le fait qu'une clé étrangère dépend d'une clé primaire, on relie la clé étrangère à la clé primaire.

Exemple :

Dans le schéma "Entreprise", l'élément "Employé" définit une clé secondaire "REFPersonne". Cette clé référence la clé primaire "IDPersonne".



L'unicité de la clé secondaire définit une contrainte unique sur un élément. Les attributs de cet élément doivent prendre les mêmes valeurs que les attributs des éléments qui portent la clé primaire.

Pour relier une clé secondaire à un élément :

1. Sélectionnez l'élément dans le navigateur de l'éditeur de schémas.
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Clés XSD**.
3. Sélectionnez "Contrainte".
4. Pour relier une contrainte existante, cliquez sur le bouton **Relier** et recherchez la contrainte.
   Pour créer une contrainte, cliquez sur le bouton **Créer** et attribuez le stéréotype "XSD Key Reference" à la contrainte.
5. Ouvrez ensuite la fenêtre de propriétés de la contrainte.
6. Cliquez sur l'onglet **Texte**.
7. Sélectionnez "Contrainte XSD" et ajoutez le texte d'unicité.
8. Cliquez sur **OK**.

Indiquez ensuite à quelle clé primaire est reliée la clé secondaire :

- Sous le nom de la clé secondaire, sélectionnez "Contrainte Référencée".
- Cliquez sur le bouton **Relier** et recherchez la clé primaire à référencer.

4

COMPLÉMENTS UML

Il est possible d'utiliser des fonctionnalités supplémentaires dans la modélisation d'un schéma grâce à l'utilisation de concepts UML .

Les points abordés ici sont :

HÉRITAGE

L'héritage est un lien entre deux classes, par lequel la classe fille hérite de tous les attributs et éléments de la classe mère.

Créer un lien d'héritage entre deux définitions de balise

Vous pouvez créer un lien d'héritage entre deux définitions de balise :

A partir du diagramme de classes :

- Glissez les deux types à relier par un lien d'*héritage* dans un diagramme de classes. Voir "Modélisation UML".

Ou dans la fenêtre de création d'un élément :

- Dans le champ **Type d'élément**, précisez le type du nouvel élément.
- Dans le champ **Type de base**, sélectionnez le type de base (l'ancêtre) du type de l'élément.
- Dans le champ **Héritage**, sélectionnez "Extension" ou "Restriction".

Héritage par extension : l'élément contient les éléments et attributs du type de base mais peut en avoir d'autres.

Héritage par restriction : l'élément a exactement le même nombre d'éléments et d'attributs que le type de base mais certains sont restreints (par la multiplicité par exemple).

Modélisation UML

- Cliquez sur le bouton **Généralisation** et reliez la classe fille à la classe mère.

Dans un schéma modélisé en UML, tout concept représenté par une classe peut hériter d'un concept de même nature :

- un type simple peut hériter d'un type simple
- un type complexe peut hériter d'un type complexe ou simple

Un type complexe est un type contenant soit des éléments (classe composée), soit des attributs, soit les deux.

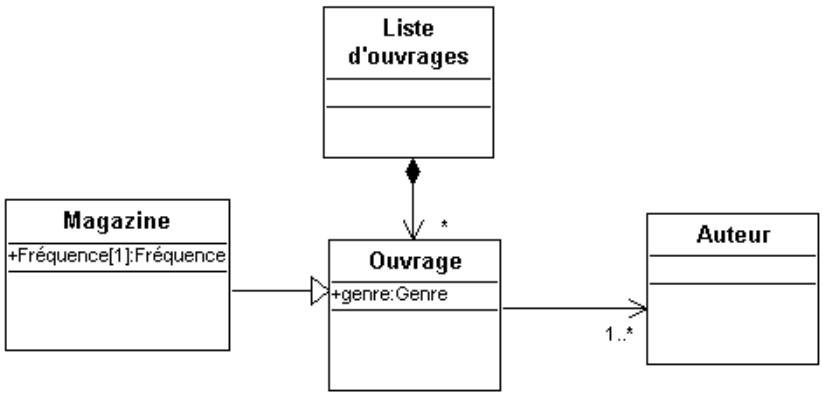
Un type simple est un type ne contenant ni sous éléments, ni attributs. A ce titre, un type complexe est nécessairement un type d'élément et un type d'attribut est forcément un type simple.

Une classe B héritant d'une classe A permet de définir des balises comportant les attributs de A et ceux de B.

De même, les éléments définis par les associations A seront ajoutés aux éléments définis à partir de B.

Exemple

La classe "Magazine" hérite de la classe "Ouvrage".



Les codes sources (XSD) générés ci-après montrent que les éléments "Magazine" disposent des attributs **Frequence** et **Genre**.

D'autre part, un attribut "Auteur" est disponible dans une balise "Magazine". Cet attribut est hérité de la classe "Ouvrage".

XSD

```
<xsd:SimpleType name="FrequenceMagazine">
   <xsd:restriction base='xsd :string'>
  <xsd:enumeration value="mensuel"/>
  <xsd:enumeration value="hebdomadaire"/>
  <xsd:enumeration value="bimensuel"/>
  <xsd:enumeration value="trimestriel"/>
</xsd:restriction>
</xsd :simpleType>
<xsd:ComplexType name="Magazine">
<xsd:complexContent>
     <xsd:extension base='ouvrage'>
     <xsd:attribute name="Frequence"
type="FrequenceMgazine" use="Required"/>
   </xsd:extension>
  </xsd:complexContent>
</xsd :complexType>
```

Utiliser les classes héritées

La notion d'héritage permet de définir des instances correspondant aux instances de la classe héritée mais disposant de spécificités.

De ce fait, si la classe B hérite de A, les instances de B peuvent être considérées comme des instances particulières de A.

En conséquence, dans un document XML, les balises A peuvent être substituées par des balises B. Pour visualiser l'héritage dans l'éditeur de Schémas :

- Cliquez sur le bouton **Affichage des classes héritées** et dépliez la classe mère pour visualiser la classe qui hérite.

Les classes qui héritent sont représentées par l'icône

> ☛ *Lors d'un héritage, tous les sous-éléments qui sont dans l'espace de nommage sont générés. Pour ne pas les générer, il faut déclarer l'élément abstrait. Voir "Créer des classes abstraites", page 73.*

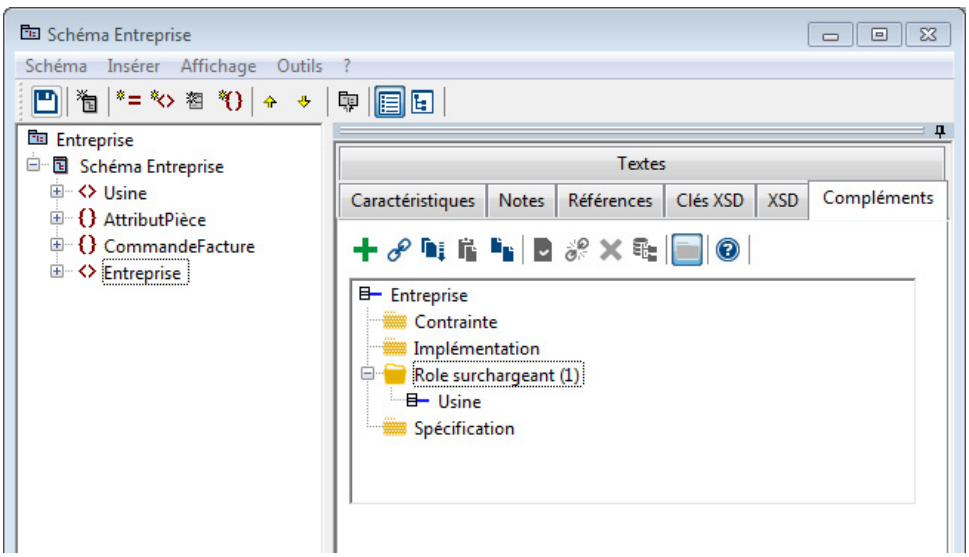
Substitution (XSD)

En XSD, il est possible de spécifier directement quels sont les éléments à substituer.

Exemple :

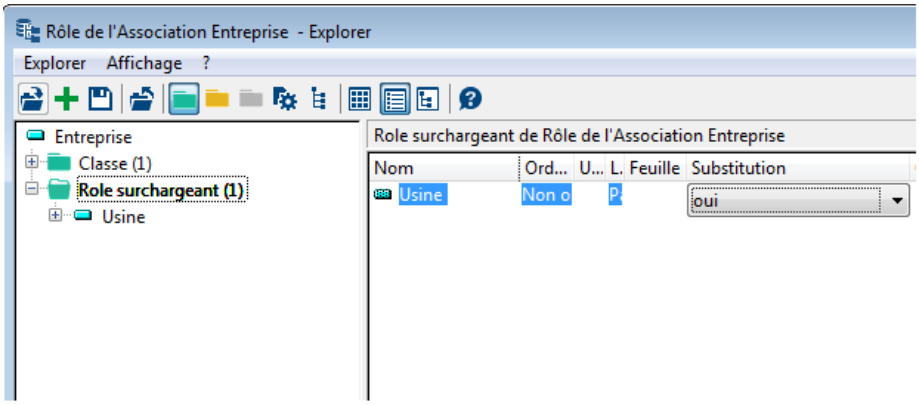
Prenons par exemple l'élément "Entreprise". Pour indiquer qu'il est substitué par l'élément "Usine :

1. Sélectionnez l'élément "Entreprise" dans le navigateur.
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Compléments**.
3. Sélectionnez "Rôle surchargeant" et cliquez sur le bouton **Relier**.
4. Dans la fenêtre de sélection, recherchez le rôle surchargeant et cliquez sur **OK**.



5. Avec le bouton droit de la souris, cliquez ensuite sur l'élément "Entreprise" et sélectionnez **Explorer**.

6. Cliquez sur "Rôle surchargeant" et dans la colonne **Substitution**, sélectionnez "oui".



COMPLÉMENTS LIÉS À L'HÉRITAGE

Lorsqu'un élément est hérité, il est doté de tous les attributs de la classe dont il hérite. Les questions suivantes se posent :

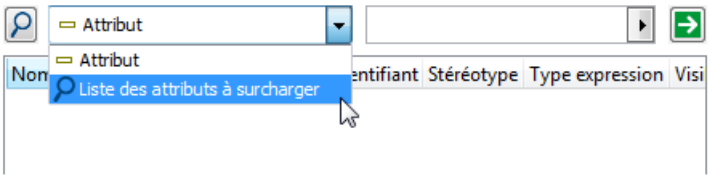
- Est-il possible de modifier les attributs ou éléments hérités (surcharge) ?
- Est-il possible d'utiliser des classes abstraites ?
- Que se passe-t-il lorsque l'élément hérité est une référence ?

Surcharger des attributs

La notion de surcharge est usuelle en programmation orientée objet mais ne s'applique généralement qu'aux méthodes d'opération. Il s'agit du mécanisme permettant de définir des propriétés supplémentaires sur un attribut hérité.

Pour *surcharger* un attribut dans l'éditeur de schéma :

1. Sélectionnez l'attribut dans le navigateur.
2. Dans la fenêtre de propriétés, cliquez sur l'onglet **Compléments**.
3. Sélectionnez le paramètre **Attribut surchargé**, et cliquez sur le bouton **Relier**.
   La fenêtre de sélection apparaît.
4. Sélectionnez **Liste des attributs à surcharger**.



5. Dans la liste de résultat, sélectionnez l'attribut et cliquez sur **relier**.

Modélisation UML

Pour surcharger un attribut dans un diagramme :

1. Affichez la fenêtre de propriétés de la classe héritée.
2. Dans l'onglet **Attribut**, sélectionnez l'attribut à surcharger puis cliquez sur le bouton **Propriétés**.
3. Procédez comme ci-dessus.

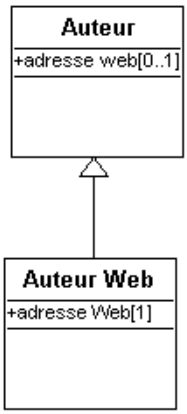
Lorsqu'une classe hérite d'une autre classe, il est possible de modifier les attributs hérités.

Cette surcharge (le fait de modifier les attributs hérités) permet de définir uniquement les spécificités de l'attribut dans la classe fille.

Toutes les propriétés de l'attribut non redéfinies dans la classe fille restent donc identiques à celles de la classe mère.

Résultat

Dans la figure ci-dessous, l'attribut "adresse Web" est surchargé dans la classe "Auteur Web".

Schéma	Diagramme
adresse Web Exemple de schémas XML::Librairie::Auteur.adresse web	

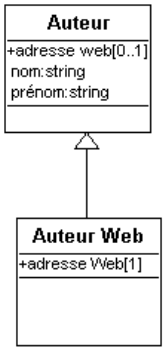
Restriction : L'attribut surchargé doit impérativement être un attribut hérité d'une classe ancêtre.

Le type de l'attribut surchargeant n'est pas défini, on considère dans ce cas que son type est celui de l'attribut surchargé (STRING).

Exemple

La classe "Auteur" dispose d'un attribut "adresse web". Cet attribut est surchargé dans la classe "Auteur Web". La différence entre la version originale et la version

surchargée de l'attribut est sa multiplicité : dans le premier cas, l'attribut est optionnel et dans la version surchargée, l'attribut est obligatoire.



XSD

```
<xsd:ComplexType name="Auteur">
  <xsd:attribute name="nom" type='xsd:string'
use="required"/>
  <xsd:attribute name="prenom" type='xsd:string'
use="required"/>
  <xsd:attribute name="adresseweb" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="AuteurWeb">
   <xsd:simpleContent>
     <xsd:restriction base='Auteur'>
       <xsd:attribute name="nom" type='xsd:string'
use="required"/>
       <xsd:attribute name="prenom" type='xsd:string'
use="required"/>
       <xsd:attribute name="adresseweb" type="xsd:string"
use='required' />
     </xsd:restriction>
  </xsd:simpleContent>
</xsd :complexType>
```

Surcharger des éléments

De même que pour un attribut, il est possible de surcharger un élément.

Cette surcharge indique que l'élément B (surchargeant) se substitue à l'élément A (surchargé) et que les propriétés de l'élément B (surchargeant) sont définies prioritairement sur A.

> **Restriction : L'élément surchargé doit impérativement être un élément hérité d'une classe ancêtre.**

Pour indiquer qu'un élément est surchargé dans l'éditeur de schéma :

1. Ouvrez la fenêtre de **Propriétés** du rôle surchargé.
2. Choisissez l'onglet **Compléments.**
3. Sélectionnez **Rôle surchargeant** et cliquez sur le bouton **Relier**.
4. Dans la fenêtre de sélection, recherchez le rôle surchargeant puis cliquez sur **Relier**.

Modélisation UML

Pour surcharger des éléments à partir d'un diagramme de classes :

1. Cliquez sur le menu **Affichage** > **Vues et détails** et cochez la case **Schéma**.
2. Cliquez sur le bouton **Lien** de la barre d'objets.
3. Dans le diagramme, cliquez sur le rôle surchargeant puis reliez-le au rôle surchargé.
4. Une fenêtre vous demande le type de lien que vous voulez créer. Sélectionnez **Rôle surchargé** puis cliquez sur **OK**.

Voir aussi "Surcharger des attributs", page 70.

Vous pouvez également surcharger des groupes.

Créer des classes abstraites

> *Les classes peuvent être définies comme abstraites. Cela signifie qu'elles ne peuvent pas être instanciées et seules des instances de classes dérivées peuvent être utilisées.*

Pour déclarer une classe abstraite :

1. Faites un clic droit sur la classe concernée et sélectionnez **Propriétés** dans le menu contextuel. La fenêtre de propriétés du type s'affiche.
2. Dans l'onglet **Caractéristiques**, cochez la propriété **Abstraite** et cliquez sur **Appliquer**.

Modélisation UML

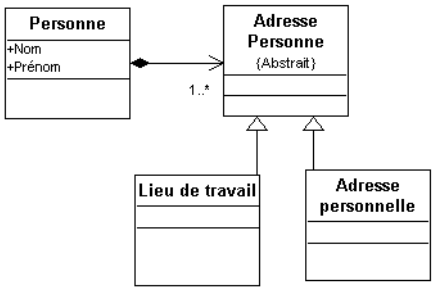
- Affichez les propriétés de la classe et procédez comme ci-dessus.

Exemple

Dans l'exemple ci-dessous, les balises définies sur la classe "Personne" disposent d'un sous-élément nommé "Adresse".

Il est lui-même défini par la classe "Adresse Personne" mais aucune instance de cette classe ne peut être utilisée car elle est déclarée abstraite.

En conséquence, seules les balises dérivées (ici "Adresse personnelle" et "Lieu de travail") peuvent être instanciées.



Exemple d'instance générée à partir du schéma :

```
<Personne Nom="Dupont" Prenom="Jean">
    <AdressePersonnelle>
    ...
    </AdressePersonnelle>
    <AdressePersonnelle>
    ...
    </AdressePersonnelle>
    <LieuDeTravail>
    ...
    </LieuDeTravail>
</Personne>
```

Héritage et référence

Fonctionnement

Le système de référence (pour plus de détails, voir "Références", page 48) met en œuvre l'insertion d'un attribut de type **ID** pour chaque balise référencée, et des attributs de type **IDREF**/**IDREFS** pour les balises se servant de la référence.

Lorsqu'une balise est référencée à partir d'une classe utilisant l'héritage, l'attribut **ID** est propagé sur les éléments hérités.

> ☛ *Les valeurs des paramètres définissant le mode de génération de l'attribut sont propagés également.*

Exemple

C'est le cas de la classe "Livre" qui hérite de la classe "Ouvrage". L'attribut identifiant de la classe "Ouvrage" est nommé "ISBN". La classe "Livre" qui hérite d'"Ouvrage" héritera de l'attribut "ISBN".

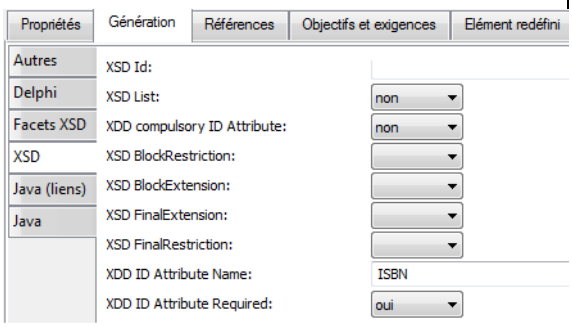
L'attribut identifiant de la classe Livre est automatiquement positionné avec cette valeur.

Ainsi la valeur des paramètres **XDD Compulsory ID Attribute**, **XDD ID Attribute Name** et **XDD ID Attribute Required** est positionnée à partir de la valeur de la classe héritée.

La classe faisant référence à "Ouvrage" fera aussi référence à "Livre".

☺ *Bien entendu, il est possible de surcharger localement n'importe lequel de ces paramètres.*

Résultat

Schéma	Document XML
	<Livre Nom="UnToutPetit-Monde" ISBN="123-456-789"/> <Ouvrage Nom="Un Ouvrage" ISBN="789-456-123"/>

CLASSES ASSOCIATIVES

Les classes associatives représentent l'association elle-même et en précisent les caractéristiques.

Pour créer une classe associative :

1. Ouvrez un **diagramme de Classes**, et glissez les deux classes associées dans le diagramme.
2. Créez une nouvelle classe.
3. A l'aide du bouton **Lien**, créez un lien entre la classe et l'association.
4. La classe d'association est reliée à l'association par un trait pointillé.

Les attributs de classes associatives sont utilisés lors de la construction d'un schéma : ils correspondent aux attributs de l'association qui ne sont utiles que dans le contexte de l'élément.

Lors de la génération des schémas, les attributs des classes associatives migrent vers la balise que la classe associée définit.

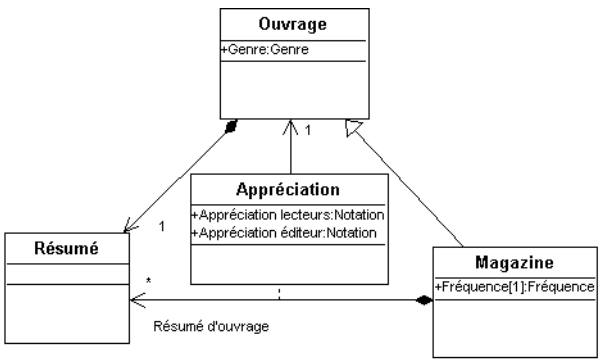
Exemple

La classe "Ouvrage" définit une balise contenant un sous-élément "Résumé".

Les balises basées sur la classe "Magazine" disposent également du résumé puisque la classe de définition hérite d'"Ouvrage".

D'autre part, les magazines peuvent contenir des résumés d'ouvrage (par exemple dans une rubrique de critique littéraire).

Dans ce contexte d'utilisation, les résumés d'ouvrages basés sur la classe "Résumé" se voient pourvus d'attributs supplémentaires : "Appréciation lecteurs", "Appréciation éditeur". Ces attributs additionnels sont modélisés par la classe associative "Appréciation".



Les schémas ci-dessous montrent la différence entre la balise "Résumé" générée pour les livres et la balise "Résumé d'ouvrage" générée pour les magazines.

XSD

```
<xsd:ComplexType name="Resume" >
    <xsd:attribute name="id" type="xsd:ID"/>
  </xsd:ComplexType>
  <xsd:ComplexType name="Resumedouvrage">
    <xsd:attribute name="id" type="xsd:id"/>
    <xsd:attribute name="Ouvrage" type="xsd:idref"
use="required"/>
    <xsd:attribute name="Appreciationlecteurs"
use="required">
      <xsd:enumeration value='Tresbon'/>
      <xsd:enumeration value='Bon'/>
      <xsd:enumeration value='Moyen'/>
      <xsd:enumeration value='Mauvais'/>
      <xsd:enumeration value='Tresmauvais'/>
     </xsd:attribute>
    <xsd:attribute name=" Appreciationediteur"
use="required">
      <xsd:enumeration value='Tresbon'/>
      <xsd:enumeration value='Bon'/>
      <xsd:enumeration value='Moyen'/>
      <xsd:enumeration value='Mauvais'/>
      <xsd:enumeration value='Tresmauvais'/>
     </xsd:attribute>
  </xsd:ComplexType>
```

Résultat

Schéma	Diagramme	Document XML
Résumé d'ouvrage (Résumé) Appréciation	Appréciation +Appréciation lecteurs:Notation +Appréciation éditeur:Notation Résumé Magazine +Fréquence[1]:Fréquence * Résumé d'ouvrage	<Magazine genre="littérature" Frequence ="Hebdomadaire" ReferenceBibliographique="125" ISBN="1" Auteur="Jacques" <Resumedouvrage Appreciationlecteur = "Bon" Appreciationediteur = "Moyen"</ ResumedOuvrage>< /Magazine>

Pour visualiser la classe associative dans l'éditeur de schémas XML :

- Cliquez sur le bouton **Affichage des types** et ouvrez l'association sur laquelle est définie la classe associative.

STÉRÉOTYPES DE CLASSE

Vous pouvez définir un des stéréotypes suivants pour une classe :

- Schema group
- Expression
- Enumération
- Structure (XSD)

Définir un stéréotype de classe

Pour définir un stéréotype :

1. Ouvrez la fenêtre de **Propriétés** de la définition de balise (ou de la classe).
2. Cliquez sur l'onglet **Caractéristiques**.
3. Dans le champ **Stéréotype** sélectionnez à l'aide de la flèche "Schema group", "Enumération" ou "Expression".

Stéréotype Schema group

*Les classes de stéréotype **Schema Group** représentent des regroupements ordonnés ou des sélections d'un ensemble d'éléments ou d'attributs.*

Les ordonnancements possibles sur un groupe sont :

All : les éléments du type peuvent apparaître aucune ou une fois et dans n'importe quel ordre. Les groupes "all" ne sont autorisés qu'au plus haut niveau d'un type et doivent être seuls. De plus, ils ne peuvent contenir que des éléments.

Sequence : les éléments sont déclarés dans l'ordre de leur définition dans le groupe.

Choice : seul un élément du groupe peut être déclaré.

Non renseigné : aucun ordre n'est pris en compte.

Ces options d'ordonnancement sont combinées aux valeurs des multiplicités.

Stéréotype Expression

Les classes de stéréotype **Expression** permettent de définir des variantes de type primitif. Elles sont basées sur les types fournis en standard (paquetage **Standard::Types::XML::XSD**).

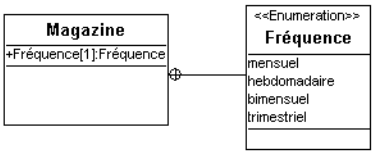
Stéréotype Enumération

Les classes de stéréotype **Enumération** permettent de définir un ensemble de valeurs. Ces classes pourront être utilisées comme type d'attributs, voire d'éléments quand le langage de schéma utilisé le permet (par exemple, pour XSD).

Ainsi, les attributs basés sur un tel type ne pourront prendre que les valeurs définies par la classe **Enumération**.

> ☛ *Il est conseillé de déclarer les types **Enumération** localement à la classe C qui définit les attributs basés sur ces types. Cette recommandation n'est valable que lorsque le type n'est pas réutilisé par d'autres attributs externes à la classe C. Ceci évite la prolifération de type au niveau du paquetage et permet de reprendre les mêmes noms quand il s'agit d'une notion similaire.*

Exemple



La classe "Magazine" dispose d'un attribut "Fréquence". Ce dernier est basé sur le type "Fréquence" de stéréotype "Enumération". Les valeurs possibles sont "mensuel", "hebdomadaire", "bimensuel" et "trimestriel".

Conformément à la recommandation précédente, la classe "Fréquence" est déclarée localement à la classe "Magazine".

XSD

```
<xsd :simpleType name='FrequenceMgazine'>
<xsd:restriction base='xsd :string'>
  <xsd:enumeration value="mensuel"/>
  <xsd:enumeration value="hebdomadaire"/>
  <xsd:enumeration value="bimensuel"/>
  <xsd:enumeration value="trimestriel"/>
</xsd:restriction>
</xsd :simpleType>
<xsd:ComplexType name="Magazine">
<xsd:complexContent>
     <xsd:extension base='ouvrage'>
     <xsd :sequence>
     <xsd:element name="Resume" type="Resume" minOccurs="1"
maxOccurs="1"/>
      < xsd:element name="Resumedouvrage"
type="Resumedouvrage" minOccurs="0"
maxOccurs="*"/>
      </xsd:sequence>
     <xsd:attribute name="Frequence"
type="FrequenceMgazine" use="Required"/>
   </xsd:extension>
  </xsd:complexContent>
</xsd :complexType>
```

Stéréotype Structure (XSD)

Les classes de stéréotype **Structure** permettent de définir des unions de types. Ce stéréotype est spécifique à XSD. Voir "Définition d'union", page 56.

CONTRÔLES

Vous avez la possibilité de vérifier la validité des schémas générés dans **MEGA**.

Les points suivants sont abordés ici :

A PROPOS DES CONTRÔLES

☛ *Dans les paragraphes qui suivent, la terminologie utilisée pour décrire les contrôles est celle d'UML. En effet l'outil de contrôle s'applique aux concepts UML.*

La fonctionnalité de contrôle recense les anomalies propres à la construction des schémas. Une phase de contrôle peut être entamée avant une génération de schéma.

Pour effectuer un contrôle :

1. Faites un clic droit sur le paquetage et sélectionnez **Contrôler** > **Description de contrôle**.
2. Sélectionnez un des contrôles dans la boîte qui s'affiche :
   - **Check Package (XML Document Definition)** : vérification de la compatibilité de la modélisation pour la génération de schémas XML (XSD).
   - **Check Package (UML Semantic)** : vérification de la compatibilité de la modélisation avec la sémantique UML.

CONTRÔLES DES SCHÉMAS XML

Les contrôles sont réalisés par type de segment. Ce niveau de contrôle recense les anomalies propres à la modélisation des schémas.

Contrôles des associations

Une association doit être binaire.

Le module de contrôle signale toutes les associations non binaires (ternaires, etc.).

Le rôle opposé à une agrégation doit être navigable.

Les éléments sont décrits par des associations binaires dont un rôle est une agrégation. Dans ce cas, le rôle opposé doit être navigable (navigabilité spécifiée à "**oui**" ou non renseignée).

Contrôles des rôles d'association

Un rôle d'association vers un type élémentaire doit être nommé.

Lorsqu'un rôle d'association n'est pas nommé, il prend automatiquement le nom de la classe à laquelle il est attaché. Dans le cas des types élémentaires (**NMTOKENS**, **STRING**), le nom du rôle doit être spécifié afin de ne pas générer des éléments avec des noms réservés.

Contrôles des classes

Un groupe ne doit hériter que de groupes

Une classe de stéréotype **Schema Group** ne peut hériter que d'autres classes du même stéréotype. En l'occurrence, l'héritage vers des classes de stéréotype **Enumération** ou **Interface** est prohibé.

Une classe non stéréotypée ne doit pas hériter de groupes

Une classe dont le stéréotype n'est pas **Schema Group** ne doit pas hériter de groupes. En revanche, les stéréotypes UML usuels comme **Frontière**, **Entité**, etc. ne sont pas contrôlés. Toutefois, ils ne devraient pas apparaître dans un schéma (sauf si le modèle de données et la structure du document sont modélisés par les mêmes classes).

Une classe ne doit avoir au maximum qu'un seul attribut **ID**.

Une balise XML ne peut disposer que d'un seul attribut XML de type **ID**.

La contrainte dans **MEGA** concerne donc les classes qui modélisent ces balises.

Une classe doit disposer au maximum d'un seul attribut de type ID.

Toutefois, cette contrainte est rendue plus complexe par le mode de création automatique des attributs de ce type. De plus, il est nécessaire de prendre en compte les attributs hérités par la classe mais non surchargés.

Le contrôle indiquera donc l'ensemble des attributs de type **ID** ainsi que l'attribut créé automatiquement quand l'option est demandée (paramètre **XDD compulsory ID Attribute**).

Contrôles des attributs

Les attributs non optionnels ne doivent pas avoir de valeur initiale

Les attributs obligatoires (multiplicité égale à **1** ou non spécifiée), ne peuvent pas disposer de valeur initiale puisqu'ils doivent toujours être spécifiés.

La cardinalité d'un attribut ne doit pas être supérieure à 1

Dans le langage XML, une balise ne peut énoncer un attribut plus d'une fois. Sa multiplicité dans **MEGA** sera donc au maximum égale à 1. En somme, les valeurs autorisées sont : **non renseigné**, **0**, **0..1** et **1**.

CONTRÔLES UML

Les contrôles UML fournis par **MEGA Designer - Development** et **MEGA for UML** correspondent à la liste définie par **UML**. Parmi eux, on trouve le contrôle de cycle sur l'héritage et la non-homonymie des attributs d'une classe ou des classes détenues par un paquetage.

COMPOSANTS

Les points suivants sont abordés ici :

GÉNÉRER UN COMPOSANT

Un composant est un regroupement d'objets que l'on veut générer.

Pour créer un composant :

1. Cliquez sur l'espace de nommage (le paquetage) du schéma et sélectionnez **Nouveau > Composant**.
2. Dans la fenêtre **Nouveau composant**, saisissez son **Nom**, et sélectionnez son **Type**, c'est-à-dire sa cible de génération.
3. Cliquez sur le bouton **Suivant**.
4. Précisez le contenu du composant (paquetages, classes, sous-composants). Il n'est pas obligatoire de le faire immédiatement car vous pouvez également le faire par la suite, par exemple dans le navigateur ou dans un diagramme de composants.
5. Cliquez sur **Suivant**.
6. Précisez le dossier de génération, ainsi que d'autres caractéristiques de la génération.
7. Cliquez sur **Suivant**.
8. La fenêtre qui apparaît vous permet de préciser le stéréotype du composant. Cliquez sur **Terminer**.

Le composant apparaît dans l'arborescence du paquetage.

STRUCTURE DES COMPOSANTS

Il existe deux types de structure d'un composant :

- le stéréotype **Fichier**
- le stéréotype **Projet**

Stéréotype Fichier

Un composant **Fichier** contient les schémas (classes) à générer et génère par défaut tous les schémas les uns à la suite des autres, dans un fichier unique.

Stéréotype Projet

Un composant **Projet** rassemble des sous-projets. Il peut inclure :

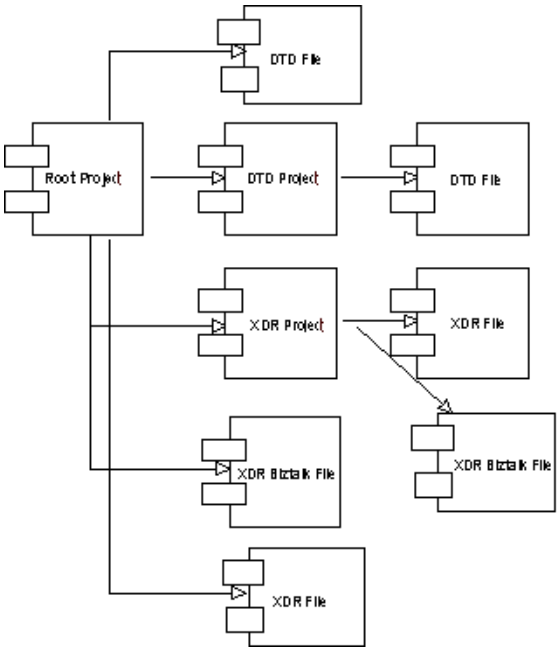
- des classes
- des paquetages
- des sous-composants (c'est à dire des composants-fichiers ou des sous-composants projets).

Dans un composant projet tous les schémas seront par défaut générés dans des fichiers séparés.

☛ *Si le composant projet contient un paquetage, un fichier par classe de type **XML Document Definition Root** du paquetage sera généré.*

☛ *Si le composant projet contient un sous-composant, un fichier rassemblant l'ensemble du code du sous-composant sera généré.*

Hiérarchie des composants disponibles



Root Project

Composant standard assurant la génération des composants fils quels qu'ils soient.

XSD Project

Génère un fichier pour chaque élément détenu par le composant. Les composants possibles sont définis dans la sous liste suivante.

- **XSD File** : génère un fichier au format XSD correspondant à la classe schéma associée au composant.

EXEMPLES DE SCHÉMAS XML COMPLETS

Les exemples donnés ici illustrent les concepts et fonctionnalités fournis dans **MEGA** pour construire un schéma XML.

Trois schémas sont présentés : "Librairie", "Personne" et "Entreprise".

PERSONNE

Schéma Personne

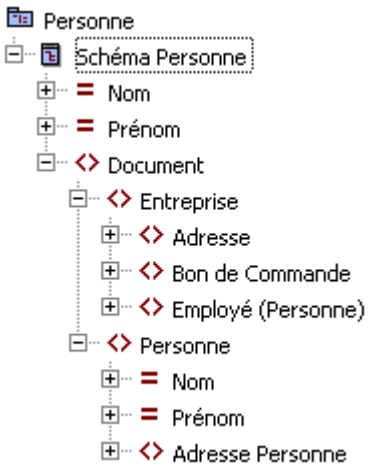
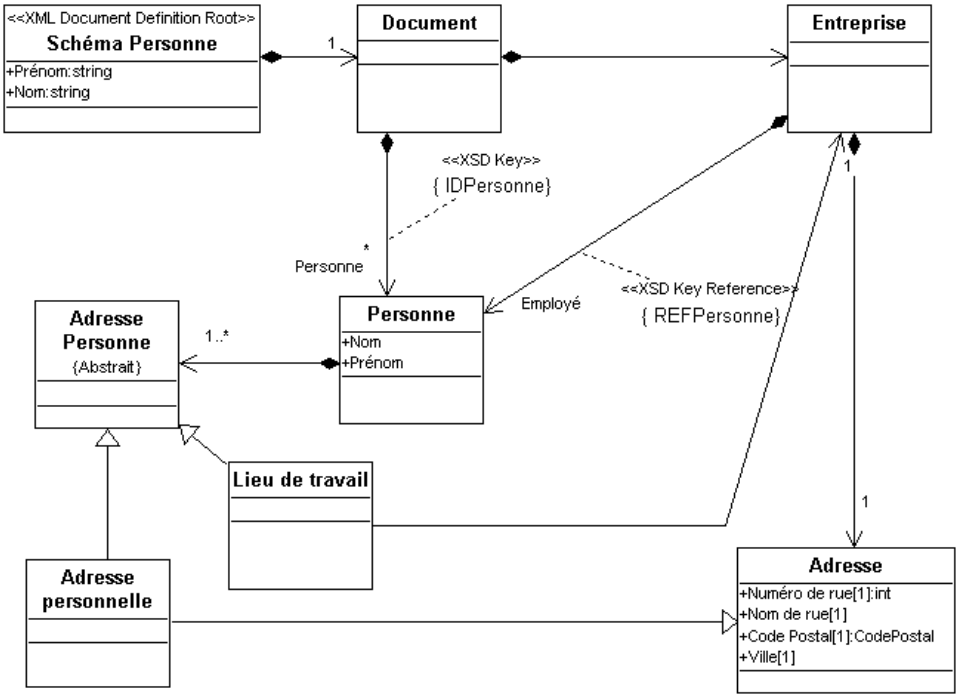


Diagramme Personne

Ce diagramme illustre la notion de schéma en général (classe de stéréotype **XML Document Definition Root**), les modélisations des éléments XML par les associations et l'utilisation de classes abstraites (Adresse Personne).



Génération XSD Schéma Personne

```
<xsd:schema targetNamespace="Personne" xmlns="Personne"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:Alias1="urn:schemas-microsoft-com:datatypes"
xmlns:Alias2="Librairie" xmlns:Alias3="Entreprise">
  <xsd:import namespace="urn:schemas-microsoft-
com:datatypes"/>
  <xsd:import namespace="Librairie"/>
  <xsd:import namespace="Entreprise" schemaLocation=""/>
  <xsd:element name="Document" type="Document"
minOccurs="1" maxOccurs="1">
  </xsd:element>
  <xsd:attribute name="Nom" type="xsd:language"
use="required">
```

```
  </xsd:attribute>
  <xsd:attribute name="Prenom" type="xsd:language"
use="required">
  </xsd:attribute>
  <xsd:simpleType name="USStateList">
    <xsd:list base="USState">
    </xsd:list>
  </xsd:simpleType>
  <xsd:simpleType name="USState">
    <xsd:restriction base="xsd:language">
      <xsd:enumeration value="PA"/>
      <xsd:enumeration value="NY"/>
      <xsd:enumeration value="CA"/>
      <xsd:enumeration value="LA"/>
      <xsd:enumeration value="AK"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="CodePostal">
    <xsd:union memberTypes="xsd:int USState">
    </xsd:union>
  </xsd:simpleType>
  <xsd:complexType name="Adresse" abstract="false"
mixed="false">
    <xsd:sequence>
      <xsd:element name="Description"
type="Alias2:Commentaire" minOccurs="1" maxOccurs="1">
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
    <xsd:attribute name="Numeroderue" type="xsd:int"
use="required">
    </xsd:attribute>
    <xsd:attribute name="CodePostal" type="CodePostal"
use="required">
    </xsd:attribute>
    <xsd:attribute name="Ville" type="xsd:anyType"
use="required">
    </xsd:attribute>
    <xsd:attribute name="Nomderue" type="xsd:anyType"
use="required">
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="AdressePersonne" abstract="true"
mixed="false">
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
```

```
  <xsd:complexType name="Lieudetravail" abstract="false"
mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="AdressePersonne">
        <xsd:attribute name="Entreprise" use="required"
type="xsd:IDREF"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Adressepersonnelle"
abstract="false" mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="AdressePersonne">
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Document" abstract="false"
mixed="false">
    <xsd:sequence>
      <xsd:element name="Entreprise"
type="Alias3:Entreprise" minOccurs="0"
maxOccurs="unbounded">
      </xsd:element>
      <xsd:element name="Personne" type="Personne"
minOccurs="0" maxOccurs="unbounded">
        <xsd:key name="IDPersonne">
          <xsd:selector xpath="Document/Personne">
  <xsd:field    xpath="@Nom"/>
  <xsd:field    xpath="@Prénom"/>
</xsd:selector>
        </xsd:key>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="Personne" abstract="false"
mixed="false">
    <xsd:sequence>
      <xsd:element name="AdressePersonne"
type="AdressePersonne" minOccurs="1" maxOccurs="unbounded">
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
    <xsd:attribute ref="Nom" use="required">
    </xsd:attribute>
    <xsd:attribute ref="Prenom" use="required">
```

```
    </xsd:attribute>
  </xsd:complexType>
</xsd:schema>
```

EXEMPLE LIBRAIRIE

Schéma Librairie

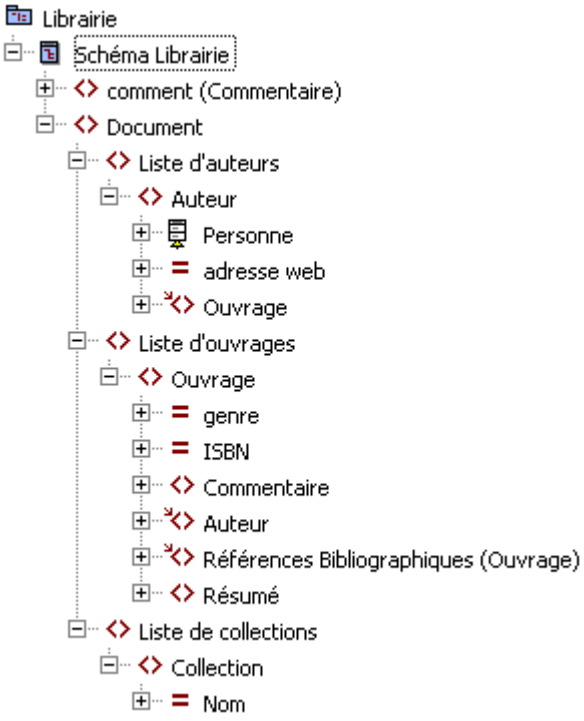


Diagramme Librairie

Ce schéma est une illustration des références bidirectionnelles (entre "Ouvrage" et "Auteur"). Il montre également la surcharge d'attributs ("Auteur web") et l'utilisation des notes.

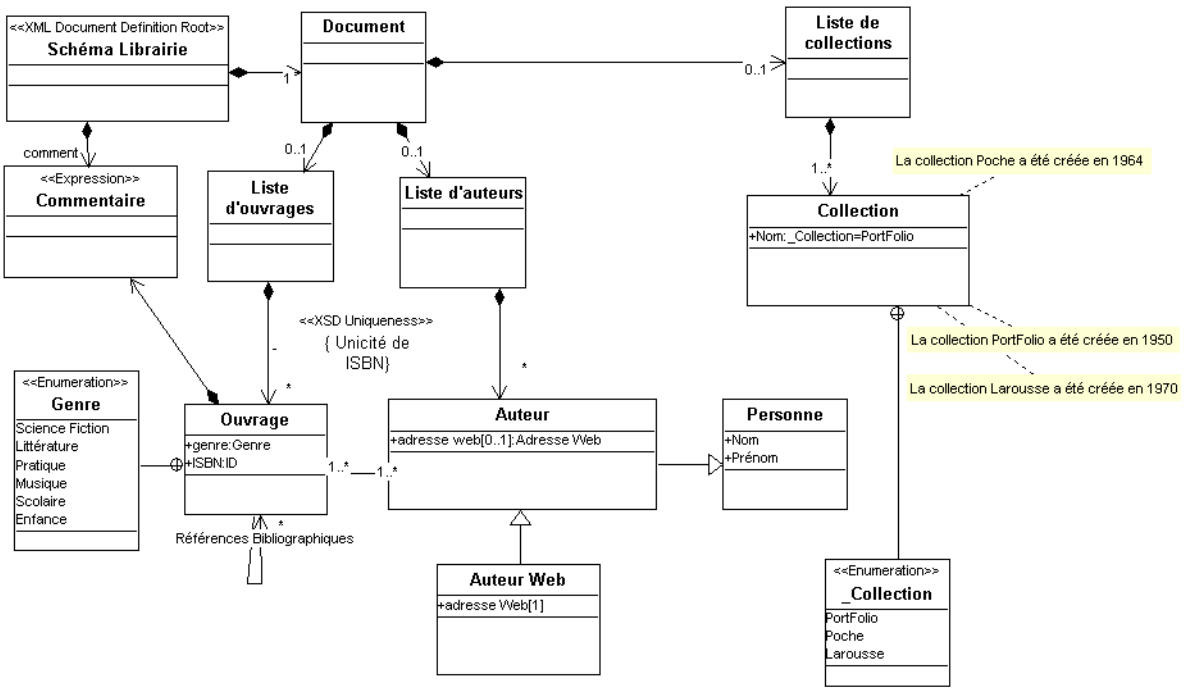
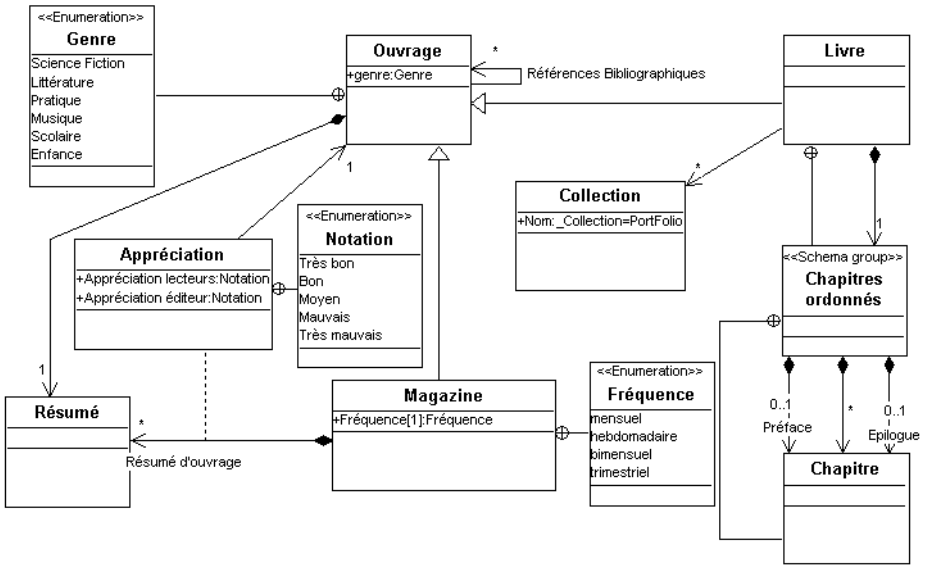


Diagramme Ouvrage

Ce diagramme montre les détails concernant la classe "Ouvrage". On y trouve les spécialisations de la classe : "Livre" et "Magazine". La notion de groupe est

également illustrée ("Chapitres ordonnés"). Enfin, un exemple de classe associative montre l'utilisation d'attributs dans un contexte spécifique (Appréciation).



Génération XSD : Schéma Librairie

```
<xsd:schema targetNamespace="Librairie" xmlns="Librairie"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:p="Personne" xmlns:d="urn:schemas-microsoft-
com:datatypes">
  <xsd:import namespace="Personne"/>
  <xsd:import namespace="Personne" schemaLocation=""/>
  <xsd:annotation>
    <xsd:documentation>Racine du document. Un document doit
commencer par un des tags reliés à partir de cette classe.</
xsd:documentation>
  </xsd:annotation>
  <xsd:element name="comment" type="Commentaire"
minOccurs="0" maxOccurs="unbounded">
  </xsd:element>
  <xsd:element name="Document" type="Document"
minOccurs="1" maxOccurs="1">
  </xsd:element>
  <xsd:simpleType name="Resume">
    <xsd:restriction base="xsd:anyType">
    </xsd:restriction>
  </xsd:simpleType>
```

```
  <xsd:simpleType name="AdresseWeb">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-z][A-Z][0-9]+@[a-z][A-Z][0-
9]"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="Commentaire">
    <xsd:restriction base="xsd:string">
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="Magazine" mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="Ouvrage">
        <xsd:sequence>
          <xsd:element name="Resumedouvrage"
type="Appreciation" minOccurs="0" maxOccurs="unbounded">
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="Frequence" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:anyType">
              <xsd:enumeration value="mensuel"/>
              <xsd:enumeration value="trimestriel"/>
              <xsd:enumeration value="bimensuel"/>
              <xsd:enumeration value="hebdomadaire"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="AuteurWeb" mixed="false">
    <xsd:simpleContent>
      <xsd:restriction base="Auteur">
        <xsd:attribute name="id" use="required"
type="xsd:ID"/>
       <xsd:attribute name="adresseWeb" type="xsd:anyType"
use="required">
        </xsd:attribute>
        <xsd:attribute ref="p:Nom" use="required">
        </xsd:attribute>
        <xsd:attribute ref="p:Prenom" use="required">
        </xsd:attribute>
   <xsd:attribute name="Ouvrage" use="required"
type="xsd:IDREFS"/>
```

```
      </xsd:restriction>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="Document" abstract="false"
mixed="false">
    <xsd:sequence>
      <xsd:element name="Listedauteurs"
type="Listedauteurs" minOccurs="0" maxOccurs="1">
      </xsd:element>
      <xsd:element name="Listedouvrages"
type="Listedouvrages" minOccurs="0" maxOccurs="1">
      </xsd:element>
      <xsd:element name="Listedecollections"
type="Listedecollections" minOccurs="0" maxOccurs="1">
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="Listedauteurs" abstract="false"
mixed="false">
    <xsd:annotation>
      <xsd:documentation>Définition de la liste des
auteurs.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
     <xsd:element name="Auteur" type="Auteur" minOccurs="0"
maxOccurs="unbounded">
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="Auteur" abstract="false"
mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="p:Personne">
        <xsd:attribute name="adresseweb" type="AdresseWeb"
use="optional">
        </xsd:attribute>
        <xsd:attribute name="Ouvrage" use="required"
type="xsd:IDREFS"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Listedouvrages" abstract="false"
mixed="false">
    <xsd:sequence>
```

```
      <xsd:element name="Ouvrage" type="Ouvrage"
minOccurs="0" maxOccurs="unbounded">
        <xsd:unique name="Unicité de ISBN">
          <xsd:selector xpath="Document/Listed'ouvrages/
Ouvrage">
  <xsd:field    xpath="@ISBN"/>
</xsd:selector>
        </xsd:unique>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="Ouvrage" abstract="false"
mixed="false">
    <xsd:sequence>
      <xsd:element ref="comment" minOccurs="0"
maxOccurs="unbounded">
      </xsd:element>
     <xsd:element name="Resume" type="Resume" minOccurs="1"
maxOccurs="1">
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="genre" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:anyType">
          <xsd:enumeration value="Musique"/>
          <xsd:enumeration value="Pratique"/>
          <xsd:enumeration value="Science Fiction"/>
          <xsd:enumeration value="Littérature"/>
          <xsd:enumeration value="Enfance"/>
          <xsd:enumeration value="Scolaire"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="ISBN" type="xsd:ID"
use="required">
    </xsd:attribute>
    <xsd:attribute name="ReferencesBibliographiques"
use="optional" type="xsd:IDREFS"/>
    <xsd:attribute name="Auteur" use="required"
type="xsd:IDREFS"/>
  </xsd:complexType>
  <xsd:complexType name="Livre" mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="Ouvrage">
        <xsd:sequence>
```

```
          <xsd:sequence>
            <xsd:element name="Preface" minOccurs="0"
maxOccurs="1">
              <xsd:simpleType name="Chapitre">
                <xsd:restriction base="xsd:anyType">
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Chapitre" minOccurs="0"
maxOccurs="unbounded">
              <xsd:simpleType name="Chapitre">
                <xsd:restriction base="xsd:anyType">
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Epilogue" minOccurs="0"
maxOccurs="1">
              <xsd:simpleType name="Chapitre">
                <xsd:restriction base="xsd:anyType">
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:sequence>
        <xsd:attribute name="Collection" use="optional"
type="xsd:IDREFS"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Collection" mixed="false">
    <xsd:annotation>
     <xsd:documentation>La collection PortFolio a été créée
en 1950</xsd:documentation>
      <xsd:documentation>La collection Poche a été créée en
1964</xsd:documentation>
      <xsd:documentation>La collection Larousse a été créée
en 1970</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
    <xsd:attribute name="Nom" fixed="PortFolio"
use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:anyType">
          <xsd:enumeration value="Poche"/>
          <xsd:enumeration value="PortFolio"/>
```

```
          <xsd:enumeration value="Larousse"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="Listedecollections" mixed="false">
    <xsd:sequence>
      <xsd:element name="Collection" type="Collection"
minOccurs="1" maxOccurs="unbounded">
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="Appreciation" mixed="false">
    <xsd:simpleContent>
      <xsd:extension base="Resume">
   <xsd:attribute name="id" use="required" type="xsd:ID"/>
        <xsd:attribute name="Appreciationlecteurs"
use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:anyType">
              <xsd:enumeration value="Très mauvais"/>
              <xsd:enumeration value="Très bon"/>
              <xsd:enumeration value="Mauvais"/>
              <xsd:enumeration value="Moyen"/>
              <xsd:enumeration value="Bon"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="Appreciationediteur"
use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:anyType">
              <xsd:enumeration value="Très mauvais"/>
              <xsd:enumeration value="Très bon"/>
              <xsd:enumeration value="Mauvais"/>
              <xsd:enumeration value="Moyen"/>
              <xsd:enumeration value="Bon"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="Ouvrage" use="required"
type="xsd:IDREF"/>
      </xsd:extension>
    </xsd:simpleContent>
```

```
  </xsd:complexType>
</xsd:schema>
```

EXEMPLE ENTREPRISE

Cet exemple illustre plus particulièrement les fonctions XSD, notamment :

- Les définitions de groupes modèles au niveau du schéma
- Les substitutions
- Les références d'éléments
- L'héritage par restriction et extension

Schéma Entreprise

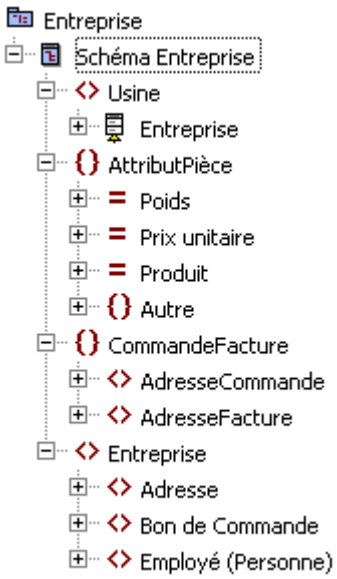
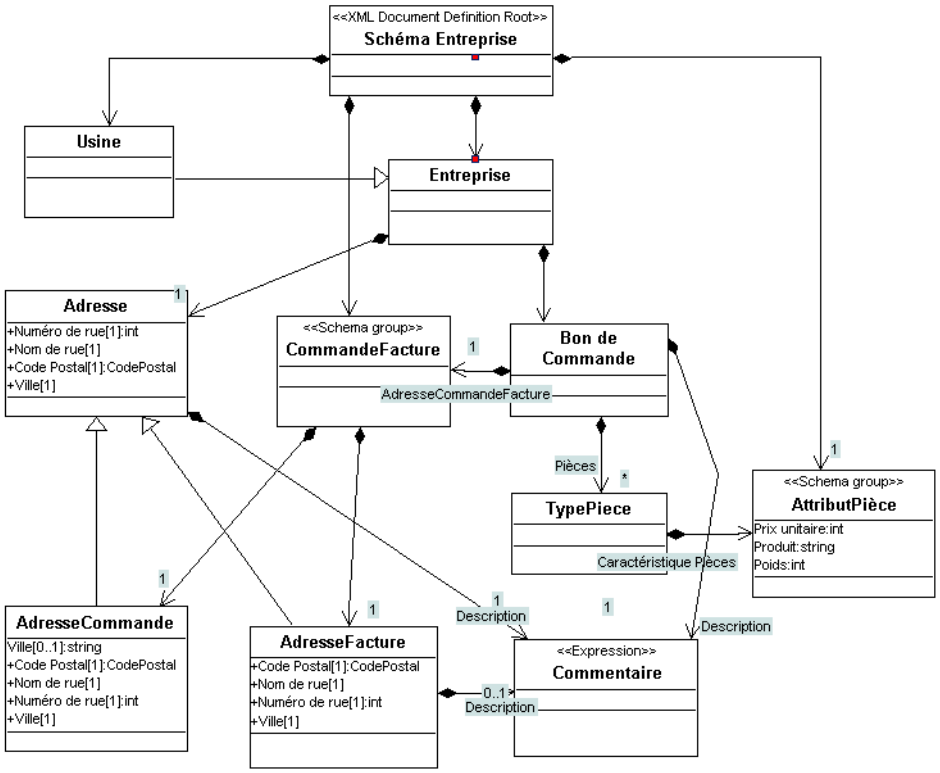


Diagramme Entreprise



Génération XSD : Schéma Entreprise

```
<xsd:schema targetNamespace="Entreprise" xmlns="Entreprise"
xmlns:l="Librairie" xmlns:Alias1="Personne"
xmlns:Alias2="urn:schemas-microsoft-com:datatypes"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import namespace="Personne"/>
  <xsd:import namespace="Librairie"/>
  <xsd:element name="Usine" type="Usine"
substitutionGroup="Entreprise" minOccurs="0"
maxOccurs="unbounded">
  </xsd:element>
  <xsd:element name="Entreprise" type="Entreprise"
minOccurs="0" maxOccurs="unbounded">
  </xsd:element>
  <xsd:group name="CommandeFacture">
```

```
    <xsd:sequence>
      <xsd:element name="AdresseCommande"
type="AdresseCommande" minOccurs="1" maxOccurs="1">
      </xsd:element>
      <xsd:element name="AdresseFacture"
type="AdresseFacture" minOccurs="1" maxOccurs="1">
      </xsd:element>
    </xsd:sequence>
  </xsd:group>
  <xsd:attributeGroup name="AttributPiece">
    <xsd:attribute name="Produit" type="xsd:string"
use="required">
    </xsd:attribute>
    <xsd:attribute name="Prixunitaire" type="xsd:int"
use="required">
    </xsd:attribute>
    <xsd:attribute name="Poids" type="xsd:int"
use="required">
    </xsd:attribute>
    <xsd:anyAttribute>
    </xsd:anyAttribute>
  </xsd:attributeGroup>
  <xsd:complexType name="TypePiece" mixed="false">
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="AdresseCommande" mixed="false">
    <xsd:simpleContent>
      <xsd:restriction base="Alias1:Adresse">
   <xsd:sequence>
         <xsd:element name="Description"
type="l:Commentaire" minOccurs="1" maxOccurs="1">
         </xsd:element>
   </xsd:sequence>

        <xsd:attribute name="id" use="required"
type="xsd;ID"/>
        <xsd:attribute name="Ville" type="xsd:anyType"
use="optional">
        </xsd:attribute>
        <xsd:attribute name="Numeroderue" type="xsd:int"
use="required">
        </xsd:attribute>
        <xsd:attribute name="CodePostal"
type="Alias1:CodePostal" use="required">
        </xsd:attribute>
```

```
        <xsd:attribute name="Nomderue" type="xsd:anyType"
use="required">
        </xsd:attribute>
      </xsd:restriction>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="AdresseFacture" mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="Alias1:Adresse">
        <xsd:sequence>
         <xsd:element name="Description" type="Commentaire"
minOccurs="0" maxOccurs="1">
          </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BondeCommande" mixed="false">
    <xsd:sequence>
      <xsd:element name="Pieces" type="TypePiece"
minOccurs="0" maxOccurs="unbounded">
      </xsd:element>
      <xsd:element name="Description" type="l:Commentaire"
minOccurs="1" maxOccurs="1">
      </xsd:element>
      <xsd:group ref="CommandeFacture"/>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="Entreprise" mixed="false">
    <xsd:sequence>
      <xsd:element name="Adresse" type="Alias1:Adresse"
minOccurs="1" maxOccurs="1">
      </xsd:element>
      <xsd:element name="Employe" type="Alias1:Personne"
minOccurs="0" maxOccurs="unbounded">
        <xsd:keyref name="REFPersonne" refer="IDPersonne">
          <xsd:selector xpath="Document/Entreprise/
Personne">
  <xsd:field    xpath="@Nom"/>
  <xsd:field    xpath="@Prénom"/>
</xsd:selector>
        </xsd:keyref>
      </xsd:element>
      <xsd:element name="BondeCommande"
type="BondeCommande" minOccurs="0" maxOccurs="unbounded">
```

```
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="Usine" mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="Entreprise">
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

GÉNÉRATION XSD

Le produit **MEGA for UML** permet de générer des fichiers XSD (XML Schema Definition) à partir de schémas XML ou de diagrammes UML construits dans **MEGA** ou ayant été rétro-générés.

Vous trouverez ici la description du code XSD généré à partir d'un schéma XML dans **MEGA**.

Afin de permettre au schéma XML de faire référence aux types **XSD**, il est nécessaire d'importer la bibliothèque correspondante. Voir "Présentation de l'éditeur de schémas", page 13.

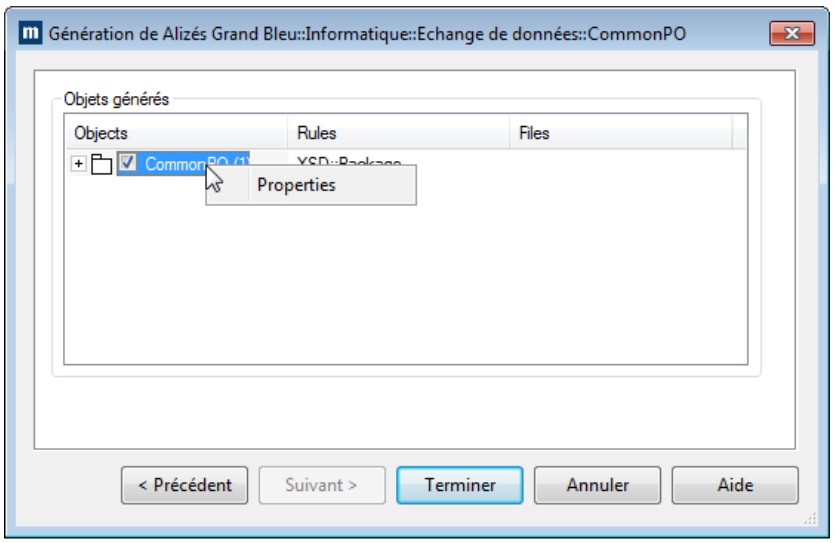
GÉNÉRER UN SCHÉMA XSD

La génération de schémas XSD est accessible à partir des objets suivants :

- UML paquetage de stéréotype "XML Document Definition"
- UML classe de stéréotype "XML Document Definition Root"
- UML composant de stéréotype "Projet"
- UML composant de stéréotype "Fichier"

Pour générer un schéma XSD :

1. Cliquez avec le bouton droit sur l'objet à générer et sélectionnez **Générer**.
2. Dans la fenêtre qui apparaît, sélectionnez **XSD** et cliquez sur **Suivant**.
3. Dans la fenêtre suivante, indiquez le dossier de génération et cliquez sur **Suivant**.
   La fenêtre affiche les objets à générer.
   Vous pouvez paramétrer la génération de ces objets en ouvrant leur fenêtre de propriétés.



4. Cliquez sur **Terminer**.

ESPACES DE NOMMAGE ET SCHÉMAS

Générer l'espace de nommage

Un espace de nommage est représenté par un paquetage de stéréotype "XML Document Definition". Les paquetages de ce type contiennent une ou plusieurs classes de stéréotype "XML Document Definition Root" et un certain nombre de classes définissant les éléments qui forment le vocabulaire de l'espace de nommage.

Le paramètre de génération "XDD urn" du paquetage correspond à l'identificateur de l'espace de nommage (urn).

> **Il est impératif que dans la hiérarchie des paquetages détenteurs d'une classe, un seul paquetage ait le stéréotype XML Document Definition. En d'autre termes, un espace de nommage ne peut détenir un autre espace de nommage ou être détenu par un autre espace de nommage.**

Autres paquetages

Vous pouvez utiliser les paquetages non interprétés comme espaces de nommage à des fins d'organisation. Vous pouvez regrouper des paquetages représentant des espaces de nommage dans un ou des paquetages qui ne sont pas de stéréotype "XML Document Definition". De même, les classes définissant le vocabulaire peuvent être "rangées" dans des paquetages inclus dans le paquetage "espace de nommage".

La classe schéma

Les schémas sont représentés par des classes de stéréotype "XML Document Definition Root".

Le vocabulaire contenu par le schéma est relié à la classe schéma par la patte "XML Document Definition Tag", ce qui permet de partitionner l'espace de nommage. Si des définitions de balise sont contenues dans le paquetage "espace de nommage" sans être reliées à une classe schéma, les définitions de balise apparaîssent dans chaque schéma de l'espace de nommage.

La classe racine sert également de point de départ pour les déclarations d'éléments ou d'attributs globaux (respectivement associés aux rôles de classes UML et aux attributs de classes UML).

Vous pouvez paramétrer une classe schéma sous l'onglet **XSD** de sa fenêtre de propriétés :

- **XSD Name** : nom de la classe schéma dans le cadre de la génération XSD. Le nom du fichier généré contenant le schéma est calculé à partir

du nom de la classe schéma (ou du paramètre "XSD Name" s'il est présent).

- **XSD AttributeFormDefault** : correspond à l'attribut "attributeFormDefault".
- **XSD BlockExtension**, **XSD BlockRestriction** et **XSD BlockSubstitution** : permettent de définir les valeurs de l'attribut "blockDefault". La valeur de l'attribut "blockDefault" est une combinaison des valeurs "extension", "restriction" et "substitution".
- **XSD ElementFormDefault** : correspond à l'attribut "elementFormDefault".
- **XSD FinalExtension** et **XSD FinalRestriction** : permettent de définir les valeurs de l'attribut "finalDefault". La valeur de l'attribut "finalDefault" est une combinaison des valeurs "extension" et "restriction".
- **XSD id** : valeur de l'attribut "id".
- **XSD version** : correspond à l'attribut "version".
- **XSD xml:lang** : correspond à l'attribut "xml:lang".
- **specificationLocation** : désigne l'emplacement du schéma. Il est notamment utilisé pour les imports et inclusions.

Sous l'onglet **Alias**, vous pouvez relier un espace de nommage à la classe schéma. Voir "Import et inclusion d'autres schémas", page 33.

Utiliser des éléments externes au schéma

Vous pouvez utiliser des éléments externes au schéma.

Ces éléments sont :

- Les élements globaux
- Les attributs globaux
- Les "attributeGroup" (définition de groupes d'attributs)
- Les groupes d'éléments (définition de groupes modèles)
- Les types simples et les types complexes

Vous pouvez également déclarer d'autres schémas et espaces de nommage dont vous souhaitez utiliser le vocabulaire. Voir "Import et inclusion d'autres schémas", page 33.

DÉFINITIONS DE TYPE

De manière générale, les classes UML servent à définir les divers éléments du vocabulaire d'un espace de nommage parmi lesquels les attributs, les éléments, les groupes modèles et les groupes d'attributs.

En dehors des classes imbriquées et des classes associatives, une classe définissant un type est en général reliée à une classe schéma par la patte "XML Document Definition Root". Sinon, elle est considérée comme appartenant globalement à l'espace de nommage et est donc générée dans tous les schémas de l'espace de nommage.

Parmi les définitions de type, XSD offre la possibilité de définir des types complexes (complexType) et des types simples (simpleType).

Une classe servant de définition de type ne doit pas être de stéréotype "XML Document Definition" (réservé aux classes schémas) ni de stéréotype "Schema group" (réservé aux classes définition de groupes).

Les classes associatives ne sont prises en comptes que sur les associations servant à préciser l'existence de sous éléments dans un type, c'est à dire sur les compositions. Dans ce cas, on considère que l'élément est du type défini par la classe associative et que celle ci dérive de la classe composante.

Paramètres de génération sur une classe type

Vous pouvez renseigner les paramètres de génération d'un type sous l'onglet **XSD** de sa fenêtre de propriétés :

- **XSD Name** : nom du type dans le cadre de la génération XSD. Si il n'est pas renseigné, c'est le nom de la classe qui est pris en compte.
- **XSD Id** : correspond à l'attribut "id".
- **XSD BlockExtension** et **XSD BlockRestriction** : permettent de déterminer la valeur de l'attribut "block". La valeur de l'attribut "block" est une combinaison des valeurs "extension" et "restriction".
- **XSD FinalExtension**, **XSD FinalRestriction**, **XSD FinalList** et **XSD FinalUnion** : permettent de déterminer la valeur de l'attribut "final". Cette valeur est une combinaison des valeurs "restriction", "list" et

"union" s'il s'agit d'un type simple, et une combinaison des valeurs "extension" et "restriction" si c'est un type complexe.

- **XSD List** : utilisé pour les types simples, ce paramètre permet de définir un "héritage par liste". Voir "Héritage par liste", page 118.
- **XSD union** : utilisé pour les types simples, ce paramètre permet de définir un "héritage par union".
- **XDD order** : indique la contrainte d'ordre sur les sous éléments.
- **XDD Text** : précise si le type contient ou non des données autres que des sous tags. Le résultat est répercuté sur l'attribut "mixed".
- **XDD compulsory ID Attribute** : ce paramètre est utilisé de concert avec **XDD ID Attribute Name** et **XDD ID Attribute Required**. Il indique la présence d'un attribut de type "ID".
- **XDD ID Attribute Name** : permet d'indiquer le nom de l'attribut de type ID généré si **XDD compulsory ID Attribute** est à "oui".
- **XDD ID Attribute Required** : permet de spécifier les options de présence de l'attribut dans le document instance.

Héritage des types

Types simples

Les types simples servent à définir les données qui sont contenues, soit dans des attributs, soit dans des balises.

```
<balise attribut="donnees"/>
<balise>donnees</balise>
```

Un type simple peut être dérivé d'un autre type simple. Si une classe dérive d'une autre classe représentant un type complexe, la classe dérivée est interprétée comme étant un type complexe.

> ☛ *Les classes de stéréotype expression sont générées comme des classes dérivant d'un type simple, dont le type de base est identifié par la classe reliée par la patte "UML Type de base de l'expression".*

Trois formes d'héritage sont possibles pour les types simples :

Héritage par restriction

Il est possible de créer un type simple à partir d'un autre type simple en imposant des restrictions (sur la taille des données, sur les valeurs possibles etc). Voir "Définition de restriction (Facets)", page 59.

Héritage par liste

Vous pouvez créer un type qui désigne une liste de valeurs d'un certain type. Voir "Définition d'un type "Liste"", page 60.

Héritage par union

Vous pouvez créer des types représentant un regroupement de différents types. Voir "Définition d'union", page 56.

Types complexes

Les types complexes définissent les contenus des balises (déclarés avec la balise "element") en termes de données, de sous-balises et d'attributs.

Comme pour les types simples, les types complexes bénéficient de l'héritage. Vous pouvez soit étendre, soit restreindre leur contenu.

Un type complexe peut dériver d'un type complexe (<complexContent> ou <simpleContent> si il y a restriction) ou d'un type simple (<simpleContent>).

Deux formes d'héritage sont possibles :

Héritage par extension

Vous pouvez ajouter des sous balises ou des attributs à un type déjà existant.

Pour étendre le contenu d'un type complexe :

1. Dans le diagramme de classes, cliquez sur le bouton **Généralisation** de la barre d'objets et reliez la classe à la classe représentant le type que vous voulez étendre.
   Le nouveau type est généré avec une balise <complexContent> ou <simpleContent> dans laquelle sont générés les <elements> ou <attributs> supplémentaires sous la balise <extension>.
2. Cliquez avec le bouton droit sur la généralisation et ouvrez ses **Propriétés**.
3. Cliquez sur l'onglet **Génération**.
4. Dans le champ **XSD Heritated by restriction**, sélectionnez "extension".

Héritage par restriction

Vous pouvez restreindre le contenu d'un type complexe. Vous pouvez par exemple changer les contraintes sur ses éléments ou attributs (comme la multiplicité).

Pour restreindre le contenu d'un type complexe :

1. Reliez par une généralisation la classe dérivée à la classe que vous voulez restreindre.
2. Ouvrez la fenêtre de propriétés de la généralisation.
3. Cliquez sur l'onglet **Génération**.
4. Dans le champ **XSD Heritated by restriction**, sélectionnez "restriction".

Au niveau du contenu de la classe dérivée, vous devez rajouter les attributs ou créer les compositions nécessaires afin de redéfinir les éléments dont les contraintes ont changé.

Pour ce qui est des attributs, reliez l'attribut redéfini à l'attribut de base par le lien "Attribut surchargé".

De même, un rôle représentant la nouvelle utilisation d'un sous élément est relié au rôle de base par la patte "rôle surchargé".

Les autres éléments n'ont pas à être ajoutés dans la classe qui sera générée en type restreint, ils seront générés automatiquement dans le nouveau type. Les éléments du type sont contenus dans une balise <restriction>, elle-même contenue sous une balise <complexContent> ou <simpleContent>.

GÉNÉRER LES GROUPES

Les groupes servent à définir des contraintes d'organisation sur les éléments contenus par un type de balise.

D'après les spécifications XSD, toutes les déclarations d'éléments doivent être effectuées dans des groupes.

En général, lors de la génération, les groupes sont déduits des classes de stéréotype "schemaGroup". Ce type de classe sert également à la représentation des groupes modèles d'éléments et d'attributs.

Dans certains cas, les groupes générés ne correspondent pas à une classe de stéréotype "schémaGroup" mais sont issus d'une classe représentant un type d'élément. La contrainte d'ordre est spécifiée par le paramètre de génération **XDD order** affiché dans la fenêtre de propriétés d'une classe.

Les différents groupes générés sont :

all

Un contenu "all" permet d'autoriser les éléments du type à apparaître aucune ou une fois et dans n'importe quel ordre. Les groupes "all" ne sont autorisés qu'au plus haut niveau d'un type et doivent être seuls. De plus, ils ne peuvent contenir que des éléments.

> ☛ *Le nombre de fois où l'élément apparaît est précisé par la multiplicité déclarée sur l'élément.*

choice

Un contenu "choice" entraîne l'apparition d'un seul élément parmi ceux déclarés dans le groupe.

sequence

Un groupe "sequence" oblige les éléments du groupe à apparaître dans l'ordre dans lequel ils sont déclarés dans le groupe.

any

Un groupe "any" est un groupe qui peut contenir n'importe quel type d'élément. Vous pouvez cependant restreindre le groupe d'éléments à un ensemble d'espaces de nommage.

Le paramètre **XSD Any** est accessible dans la fenêtre de propriétés du groupe. Il a comme valeurs possibles :

- "any" : définit un groupe de n'importe quel type d'élément.
- "anyAttribute" : définit un groupe de n'importe quel type d'attribut.

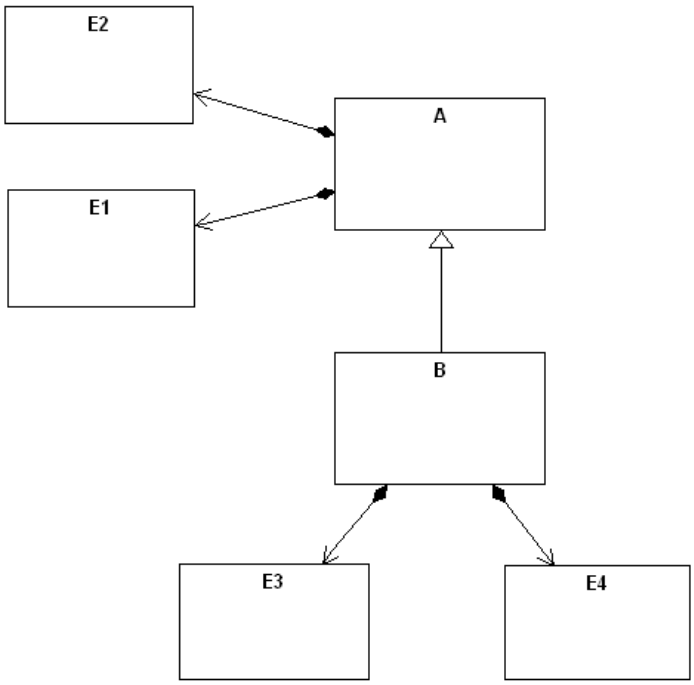
La manière dont les groupes sont générés varie selon les cas suivants :

- Si le groupe contient des sous-éléments, le groupe est généré suivant l'ordre défini ("sequence", "choice" ou "all").
- Si le groupe ne contient pas de sous-élément, il est généré suivant ce que vous avez défini sous **XSD Any** ("any" ou "anyAttribute"). Si rien n'est précisé sous **XSD Any**, rien n'est généré.

Groupes hérités

Il se peut qu'un type B contenant un groupe hérite d'un type A contenant un groupe du même type. Le schéma XML généré associé spécifiera la notion d'héritage.

Cependant, dans le document instance résolu par le schéma, l'élément de type <B> ne contiendra pas un groupe avec la concaténation des éléments de A et de B mais deux groupes : celui défini dans A puis celui défini dans B.




```
<xsd:complexType name="A" mixed="false">
      <xsd:sequence>
         <xsd:element name="E1" type="E1" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="E2" type="E2" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID" use="required"/
>
   </xsd:complexType>
   <xsd:complexType name="B" mixed="false">
      <xsd:complexContent mixed="false">
         <xsd:extension base="A">
            <xsd:sequence>
               <xsd:element name="E3" type="E3" minOccurs="0"
maxOccurs="unbounded"/>
               <xsd:element name="E4" type="E4" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
```

```
</xsd:complexType>
```

Pour n'obtenir qu'un seul groupe généré, vous devez définir un type B qui n'hérite pas du type A mais auquel vous rajoutez les attributs ou éléments de A.

DÉCLARATION D'ÉLÉMENTS

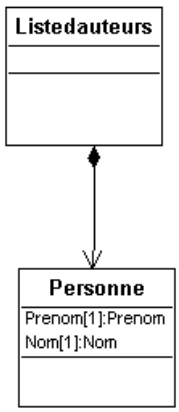
Les types complexes permettent de définir des balises susceptibles de contenir d'autres balises. Un certain nombre de contraintes permettent de préciser :

- le nombre de balises d'un type pouvant être contenues (multiplicité)
- si l'élément est abstrait
- une valeur que doit prendre la balise
- etc.

Modélisation des éléments

Les balises contenues dans un type de balises sont déduites des associations composées entre la classe représentant le type contenant (classe composée) et les classes définissant les balises contenues (classes composantes). Les caractéristiques de la déclaration sont supportées par le rôle de l'association du côté du type de balise contenu.

Exemple




```
<XSD:complexType name="Listedauteurs">
  <XSD:element name="Personne" type="Personne"/>
</XSD:complexType>
```

A partir de l'éditeur de schémas, vous pouvez renseigner sur le rôle les paramètres de génération suivants :

Onglet **XSD** :

- **XSD Name** : nom de la balise dans le cadre de la génération XSD. S'il n'est pas renseigné, c'est le nom du rôle qui est pris en compte. Dans le cas où le rôle n'a pas de nom, on prend le nom de la classe reliée au rôle.
- **XSD Id** : identifiant de la balise <element>.
- **XSD BlockExtension**, **XSD BlockRestriction** et **XSD BlockSubstitution** : ces paramètres permettent de déterminer la valeur de l'attribut "block". La valeur de l'attribut "block" est une combinaison des valeurs "extension", "restriction" et "substitution".
- **XSD FinalExtension** et **XSD FinalRestriction** : ces paramètres permettent de déterminer la valeur de l'attribut "final". Cette valeur est une combinaison des valeurs "extension" et "restriction".
- **XSD Form** : valeur de l'attribut "form".
- **XSD Value** : renseigne la valeur de l'élément. Ce paramètre est utilisé conjointement avec l'attribut "Modifiable".
  - Si "Modifiable" = "non", un attribut "fixed" est généré. Sa valeur est correspond au paramètre XSD Value.
  - Si "Modifiable" = "oui", un attribut "default" prenant cette valeur est généré.
- **XSD Abstract** : valeur de l'attribut "abstract".
- **XSD Nillable** : valeur de l'attribut "nillable".

Onglet **Caractéristiques** :

- **Multiplicité** : renseigne le nombre de fois minimal (attribut "minOccurs") et maximal (attribut "maxOccurs") que peut apparaître la balise dans la balise contenant.
- **Modifiable** : lorsque cet attribut est positionné à "non", il en résulte un attribut "fixed" dans la balise <element> (si toutefois une valeur est donnée au paramètre XSD Value).

Clés

Les clés mettent en œuvre les concepts d'unicité (clé primaire) et de référencement d'un élément.

Dans un document XSD, les clés permettent de spécifier que des éléments d'une balise (par exemple un attribut) doivent avoir la même valeur que des éléments d'une autre balise dans le document instance.

Les clés sont formalisées dans **MEGA** par des "Contraintes" reliées au rôle de classe. Le nom de la clé générée est copié sur celui de la contrainte. Le contenu de la balise <key> généré est identique au texte contenu dans l'attribut "XSD constraint" de la contrainte. Ce texte contient la définition de la clé et doit être conforme aux spécifications XSD (utilisation des balises <selector> et <field>). Voir "XSD : Les contraintes", page 61.

Parmi les clés on trouve celles qui donnent lieu à la génération de balises :

<unique>

Ce tag est généré à partir d'une contrainte dont le stéréotype est "XSD uniqueness". Cette contrainte signifie qu'il ne peut y avoir deux balises (contenant la clef) dont les éléments définis dans la clé ont la même valeur. Voir "Unicité en XSD", page 61.

<key>

Cette clé a les mêmes propriétés que <unique>. De plus, elle peut servir de référence pour une clé <keyref>. La contrainte représentant cette clé a pour stéréotype "XSD Key". Voir "Clé primaire, Clé étrangère en XSD", page 62.

<keyref>

Il s'agit d'une contrainte de stéréotype "XSD Key Reference". Voir "Clé primaire, Clé étrangère en XSD", page 62.

Référencer des éléments

Il est possible de définir un élément au niveau du schéma. Cet élément peut ensuite être référencé au sein d'une définition de balise. Le but est d'éviter la redondance de définition. Voir "Référence à un élément (Schéma XML)", page 52.

Substitution

Il est possible de déclarer qu'un élément (dénommé élément tête) peut se substituer à un autre dans le document instance. Il s'agit d'un "substitution group". L'élément substitué doit être déclaré au niveau global.

Pour créer la substitution :

1. Dans le diagramme de classes, créez un élément du type de l'élément tête ou d'un type dérivé.
2. Reliez le rôle de l'élément substituant à l'élément substitué. Il s'agit d'un lien de type "Rôle surchargé".
3. Ouvrez la fenêtre de propriétés du lien.
4. Cliquez sur l'onglet **Génération** puis le sous-onglet **XSD**.
5. Dans le champ **IsSubstituableTo**, sélectionnez "oui".

Surcharge dans le cadre de l'héritage par restriction

Lorsqu'un type dérive par restriction, vous pouvez redéfinir des éléments contenus dans le type de base. Vous pouvez par exemple réduire la multiplicité.

Pour surcharger un élément :

1. Dans le diagramme de classes, créez une composition entre le type dérivé et le type de l'élément surchargé.

2. Reliez le rôle représentant l'élément surchargeant à celui surchargé. Il s'agit d'un lien de type "Rôle surchargé".
3. Ouvrez la fenêtre de propriétés du lien.
4. Cliquez sur l'onglet **Génération** puis le sous-onglet **XSD**.
5. Dans le champ **IsSubstituableTo**, sélectionnez "non".

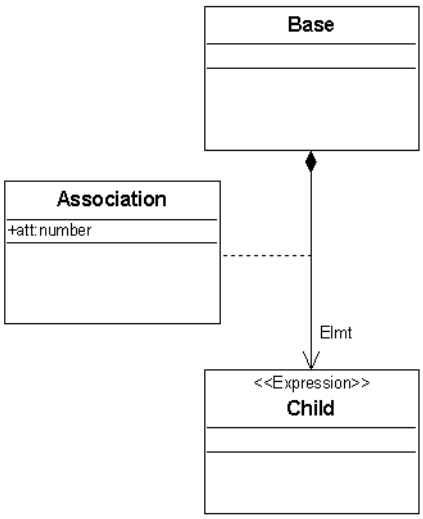
Dans tous les cas, l'élément généré dans le type dérivé aura le même nom que celui surchargé.

Classes associatives

Les classes associatives ne sont prises en compte que sur les compositions. Elles permettent de créer des éléments dont le type est étendu par rapport à celui de la classe composante. Ce type étendu n'est valable que dans le cadre de cette relation.

Dans le schéma généré, cela se traduit par la création d'un élément dont le type est dérivé de la classe composante (comme si il existait une généralisation entre la classe composante et la classe associative).

Exemple




```
<xsd:complexType name="Base" mixed="false">
    <xsd:sequence>
      <xsd:element name="Elmt" type="Association"
minOccurs="0" maxOccurs="unbounded">
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="Child">
    <xsd:restriction base="xsd:NMTOKENS">
    </xsd:restriction>
```

```
  </xsd:simpleType>
  <xsd:complexType name="Association" abstract="false"
mixed="false">
    <xsd:simpleContent>
      <xsd:extension base="Child">
        <xsd:attribute name="att" type="d:number"
use="required">
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

DÉCLARATION D'ATTRIBUTS

Les déclarations d'attributs indiquent la présence d'attributs dans un type de balise. De même que pour les déclarations d'éléments, on peut appliquer des contraintes sur les attributs :

- présence optionnelle ou non
- valeur par défaut
- etc.

Modélisation des attributs

Les attributs UML des classes correspondent aux attributs des types de balise. Pour ajouter une déclaration d'attribut dans un type de balise, vous devez créer un attribut dans la classe. Le type de l'attribut de la balise est déduit du type de l'attribut UML.

L'éditeur de schémas vous permet de paramétrer les déclarations d'attributs.

❱ Sélectionnez l'attribut dans le navigateur. Sa fenêtre de propriétés s'affiche dans la partie droite de l'éditeur.

Onglet XSD :

- **XSD Name** : nom de l'attribut dans le cadre de la génération XSD. Si ce paramètre n'est pas renseigné, c'est le nom de l'attribut qui est pris pour la génération.
- **XSD Id** : permet d'identifier la balise <attribute>.
- **XSD Form** : donne la valeur de l'attribut "form".
- **XDD Attribut UML en tant qu'élément XML** : permet de spécifier si les attributs UML sont générés systématiquement en éléments XML ou s'ils sont générés suivant leur type de base. Il existe deux options possibles :
  - **Déterminé à partir du type de base** : si l'attribut est de type simple, un attribut XML est généré. Si l'attribut est de type complexe, c'est un élément XML qui est généré.
  - **Forcé en tant qu'élément** : quelque soit le type de base, l'attribut UML est généré en tant qu'élément XML.

Onglet Caractéristiques :

- **Multiplicité** : détermine la valeur de l'attribut "use" de la déclaration d'attribut. La multiplicité peut prendre les valeurs "0", "0..1" ou "1". Par défaut la multiplicité est "1".
  Si multiplicité = "0", "use" = "prohibited".
  Si multiplicité = "0..1", "use" = "optional".
  Si multiplicité = "1", "use" = "required".
- **Modifiable** : lorsque cet attribut est positionné à "non", il en résulte un attribut "fixed" dans la balise <attribute> (si toutefois une valeur est donnée au paramètre valeur initiale).
- **Valeur initiale** : si "Modifiable" est à "non", l'attribut "fixed" est généré avec la valeur contenue dans "Valeur initiale". Sinon, c'est un attribut "default" qui est généré avec pour valeur celle de "Valeur initiale".

Références des attributs

De même que vous pouvez faire des déclarations d'éléments globaux, vous pouvez déclarer des attributs au niveau du schéma. Ces déclarations sont utilisées dans les types par référence.

```
<XSD:attribute name="globAttribute" type="globType"/>

<XSD:complexType name="cType">
  <XSD:attribute ref="globAttribute"/>
</XSD:complexType>
```

Voir "Référence à un attribut (Schéma XML)", page 53.

Attributs de type ID

Les attributs de types ID permettent d'identifier les éléments de manière unique dans tout le document.

Par défaut, les types complexes sont générés avec des attributs de types "ID". Cela provient du fait que les classes ont le paramètre de génération "XDD compulsory ID Attribute" affecté à "oui". Ce paramètre entraîne la génération d'un attribut de type "ID" dont le nom est spécifié par le paramètre "XDD ID Attribute Name" et l'option d'utilisation par le paramètre "XDD ID Attribute Required".

> ☛ *Vous pouvez modifier les paramètres de toutes les classes d'un paquetage en modifiant les paramètres du paquetage en question.*

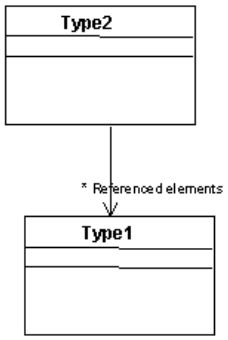
Lorsqu'un attribut UML de type expression "ID" est présent sur la classe, les paramètres précisés sous l'onglet **Génération** ne sont pas pris en compte. Il ne doit pas y avoir plus d'un attribut de type "ID" sur une classe.

Associations non composées

Les associations "simples" (sans composition) ne sont pas interprétées comme des déclarations d'élément. Lorsqu'une association relie deux classes, les rôles navigables indiquent que la classe opposée au rôle navigable référence la classe adjacente. Cela se traduit par la création d'un attribut de type "IDREF" dans le type correspondant à la classe "référençante".

Le nom de l'attribut généré est tiré du nom du rôle. De plus, la multiplicité du rôle oriente la manière dont l'attribut est généré :

- Multiplicité = "0", "use" = "prohibited"
- Multiplicité = "0..1", "use" = "optional"
- Multiplicité = "1", "use" = "required"
- Multiplicité > 1, le type de l'attribut est "IDREFS"




```
<xsd:complexType name="Type2" mixed="false">
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
   <xsd:attribute name="Referencedelements" use="optional"
type="xsd:IDREFS"/>
  </xsd:complexType>
```

GROUPES MODÈLES

Dans le cadre de la génération XSD dans **MEGA**, les classes UML servent à élaborer des contenus en précisant des contraintes d'ordonnancement lorsqu'il s'agit d'un contenu en termes de sous éléments ou des regroupement d'attributs si la classe contient des attributs.

Lorsque la classe est utilisée pour construire un type de balise, ce contenu décrit les sous balises et les sous attributs pouvant être ajoutés dans les balises de ce type.

D'autre part, vous pouvez définir des contenus qui ne donnent pas lieu à des balises dans le document instance. Par contre, ces contenus sont réutilisables dans les définitions de type de balise. Pour cela, vous devez utiliser les classes de stéréotype "schema group". Voir "Générer les groupes", page 120.

Créer et utiliser un groupe modèle

Les classes groupes modèles doivent être des composantes de la classe schéma et avoir le stéréotype "schema group".

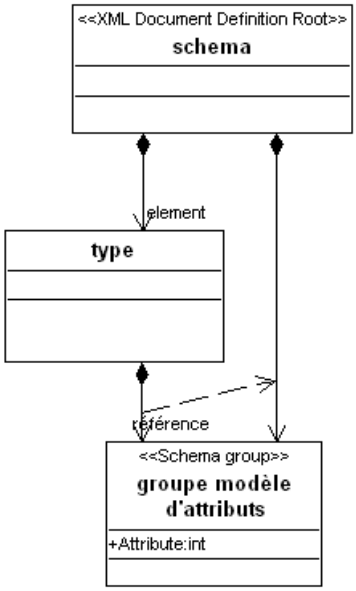
Afin d'utiliser un groupe modèle dans un type (ou dans un autre groupe), vous devez créer une composition entre le type (ou le groupe) et la classe groupe modèle. Le rôle du groupe modèle (rôle non composé) contient les informations relatives à cette utilisation (multiplicité par exemple).

Afin de pouvoir utiliser le groupe modèle dans le schéma XSD, vous devez y faire référence.

```
<xsd:group name="modelGroup">
  <xsd:choice>
    <xsd:element name="elmt" type="type1" minOccurs="1"
maxOccurs="unbounded"/>
    <xsd:element name="anotherElmt" type="anotherType"/>
  </xsd:choice>
</xsd:group>
<xsd:complexType name="type2">
  <xsd:sequence>
    <xsd:group ref="modelGroup" minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

Pour référencer un groupe modèle :

- Reliez le rôle de la composition {type, groupe modèle} à celui de la composition {schema, groupe modèle}. Il s'agit d'un lien de type "élément référencé".




```
<?xml version='1.0' encoding='ISO-8859-1'?>
<xsd:schema xmlns="" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">
  <xsd:element name="element" type="type" minOccurs="0"
maxOccurs="unbounded">
  </xsd:element>
  <xsd:attributeGroup name="groupemodeledattributs">
    <xsd:attribute name="Attribut" type="xsd:int"
use="required">
    </xsd:attribute>
  </xsd:attributeGroup>
  <xsd:complexType name="type" mixed="false">
    <xsd:attribute name="id" use="required" type="xsd:ID"/>
    <xsd:attributeGroup ref="groupemodeledattributs"/>
  </xsd:complexType>
</xsd:schema>
```

Les classes de stéréotype "schema group" servent à modéliser tant les groupes modèles d'éléments que les groupes modèles d'attributs. Une classe groupe modèle représente soit un groupe d'éléments soit un groupe d'attributs, l'un excluant l'autre.

Dans le cas où une classe "schema group" contient un ou des attributs, elle est considérée comme groupe d'attribut. Cela donne donc lieu à un groupe modèle d'attribut dont la balise dans le schéma XSD est générée sous la forme <attributeGroup> (de même que les références dans les types).

NOTES ET COMMENTAIRES

Sur les différents concepts UML utilisés pour la modélisation des schémas dans **MEGA**, vous trouverez un attribut texte "commentaire". De même, il est possible de relier des notes aux objets correspondant à ces concepts. Ces notes peuvent être de type "spécification technique" ou de type "documentation".

Les commentaires et les notes sont générés sous forme de balises <annotation> contenant des balises <documentation> ou <appinfo>.

Exemple

```
<xsd:annotation>
  <xsd:documentation>
    blabla1
  </xsd:documentation>
  <xsd:appinfo>
    blabla2
  </xsd:appinfo>
</xsd:annotation>
```

Les attributs commentaires sont générés comme balise <documentation>.

Les notes qui n'ont pas le stéréotype "spécification technique" sont générées comme balise <documentation>.

Les notes de type "spécification technique" sont générées comme balise <appinfo>.

Vous pouvez relier aux notes une "référence externe" par la patte "uri Reference". Cette référence externe sert à préciser l'adresse d'un élément complétant la note. Cette localisation est générée dans l'attribut "source" des balises <appinfo> et <documentation>.

Les objets dont les commentaires et notes sont pris en compte sont :

- les classes
- les attributs
- les rôles
- les contraintes

RÉTRO-GÉNÉRATION XSD

Vous pouvez importer un fichier XSD dans une base.

Vous trouverez ici une description expliquant comment sont rétro-générés les schémas XSD et comment ils sont modélisés dans **MEGA**.

IMPORTER UN SCHÉMA XSD DANS MEGA

Pour importer un schéma XSD dans **MEGA** :

- Cliquez sur le menu **Fichier** > **Importer** > **XSD**.

Dans la fenêtre qui apparaît, indiquez :

- Les **fichiers** XSD à rétro-générer. En général, les fichiers ont une extension (.xml) ou (.xsd).
- Le **paquetage détenteur** , autrement dit le paquetage dans lequel vous voulez rétro-générer. Si vous désirez créer une arborescence de paquetages, séparez les noms des paquetages par " :: ". Par exemple, pour remonter les informations dans le paquetage "Commandes", contenu dans le paquetage "XSD", écrivez : "XSD ::Commandes". Le paquetage associé à l'espace de nommage du fichier en cours de rétro-génération aura pour nom le targetNamespace du schéma et appartiendra à ce paquetage détenteur.
- L' **Urn** : une fois les schémas sélectionnés, s'ils ont le même espace de nommage, l'urn est automatiquement renseignée avec le targetNamespace des schémas sélectionnés. Cette valeur ne peut être modifiée.

  ☛ *Si les schémas ont été récupérés sur le Web, leur urn est probablement une adresse http (et non pas le chemin complet du schéma). Si les schémas n'ont pas le même espace de nommage, l'urn est grisée et vide.*
- Le **paquetage de référence**. Une fois la rétro-génération effectuée, tous les types des schémas référencés mais non trouvés seront stockés dans ce paquetage.
- **Conservation du fichier MGR** : lorsque vous cochez ce champ, le fichier mgr et le fichier xmgr de rétro-génération sont sauvegardés dans le répertoire temporaire de la machine.

MODÉLISATION DES BALISES DU SCHÉMA XSD

Le schéma XSD est composé de 20 balises importantes. Chacune de ces balises présente une modélisation UML.

Schema

La balise "schema" est la racine du document XML. Elle définit la structure du schéma XSD.

Le schéma est modélisé par une classe de stéréotype "XML Document Definition Root".

Espace de nommage

Son espace de nommage est modélisé par un paquetage UML de stéréotype "XML Document Definition". Ce paquetage a pour nom la valeur de l'attribut "targetNamespace" (le nom peut être modifié). Il contient l'ensemble des schémas le définissant comme espace de nommage. L'espace de nommage est caractérisé de manière unique par son urn. L'urn est modélisé par l'attribut "XDD Urn" qui se trouve dans l'onglet de génération **XML**.

Il se peut qu'un schéma n'ait pas d'espace de nommage. Dans ce cas, le schéma est rétro-généré dans le paquetage de stéréotype "XML Document Definition" n'ayant pas d'urn. Si le paquetage n'existe pas, il est créé et porte le nom de "NoTargetNamespace".

Attributs de "schema"

- **Nom** : le schéma XSD n'a pas de nom. Le nom de la classe devient donc le nom du fichier contenant le schéma. Pour générer un nom différent de celui de la classe, indiquez le nouveau nom dans le paramètre "XSD Name".
- **AttributFormDefault** : est modélisé par le paramètre de génération **XSD AttributeFormDefault**. Il prend la valeur "qualifié" ou "non qualifié".
- **ElementFormDefault** : est modélisé par le paramètre de génération X**SD ElementFormDefault**. Il prend la valeur "qualifié" ou "non qualifié".
- **BlockDefault** : est modélisé par trois paramètres de génération **XSD BlockExtension**, **XSD BlockRestriction**, et **XSD BlockSubstitution**. Chacun de ces paramètres prend la valeur "oui" ou "non" suivant que le schéma définit un blockDefault ayant pour valeur extension, restriction, substitution ou pas.
- **FinalDefault** : est modélisé par les deux paramètres de génération **XSD FinalExtension** et **XSD FinalRestriction**. Chacun de ces paramètres

prend la valeur "oui" ou "non" suivant que le schéma définit un finalDefault ayant pour valeur extension, restriction ou pas.

- **TargetNamespace** : est modélisé par le paquetage détenteur du schéma.
- **Version** : est modélisé par l'attribut **XSD Version**.
- **Xml :lang** : est modélisé par l'attribut **XSD xml :lang**. Ce paramètre a des valeurs prédéfinies comme "fr", "de", "en-GB", "en-US". Cette liste peut être étendue.
- **SchemaLocation** : cet attribut définit l'emplacement physique du schéma. Il est modélisé par le paramètre **XSD SpecificationLocation**.

Include - Import

Include

La balise "Include" permet à un schéma d'inclure les types, éléments, attributs et groupes d'un schéma appartenant au même namespace. Cela permet de réutiliser des définitions existantes et d'éviter les redondances. Ainsi, on peut avoir une grande modularité entre les schémas utilisés.

La balise "Include" est modélisée par le lien "UML Reference Class-to-Class" entre le schéma incluant et le schéma inclus.

Attributs de "Include"

- **Id** : est modélisé par l'attribut de lien **XSD Id**.
- **SchemaLocation** : est modélisé par le paramètre de génération **XSD SpecificationLocation** qui est localisé sur la classe référencée.
- **Reference** : il s'agit d'un attribut **MEGA** qui permet de savoir si le schéma consulté est inclus ou importé. Il est représenté par l'attribut **XSD Schema Reference**.

Import

La balise "Import" permet à un schéma d'importer les types, éléments, attributs et groupes d'un schéma appartenant à un autre espace de nommage. Cela permet de réutiliser des définitions existantes et d'éviter les redondances.

La balise "Import" peut être modélisée de deux façons :

- Soit par le lien "UML Reference Class-to-Class" entre deux classes. Dans ce cas, le schéma courant importe un unique schéma.
- Soit par le lien "UML Reference Class-to-Package" entre une classe et un paquetage. Le schéma courant importe alors un espace de nommage complet.

Attributs de 'Import"

- **Id** : est modélisé par l'attribut de lien **XSD Id**.
- **SchemaLocation** : est modélisé par le paramètre de génération **XSD SpecificationLocation** qui est localisé sur la classe référencée. Cet attribut n'est utilisé que dans le cadre d'un import de schéma.
- **Namespace** : est modélisé par l'attribut **XDD Urn** du paquetage détenteur du schéma importé ou du paquetage importé.
- **Reference** : il s'agit d'un attribut **MEGA** qui permet de savoir si le schéma consulté est inclus ou importé. Il est représenté par l'attribut **XSD Schema Reference**. Cet attribut n'est utilisé que dans le cadre d'un import ou d'une inclusion de schéma.

SimpleType

La balise "SimpleType" définit une définition de balise ou d'attribut XML, autrement dit, elle définit la structure d'une balise ou d'un attribut. Cette structure peut ensuite être réutilisée par plusieurs balises ou attributs. La structure est simple, c'est-à-dire qu'elle ne peut contenir ni d'attribut, ni de sous-tag. Par contre, elle peut représenter une liste, une union de types ou encore imposer des restrictions sur des types simples déjà existants.

La balise "SimpleType" est modélisée par une classe UML.

Attributs de "SimpleType"

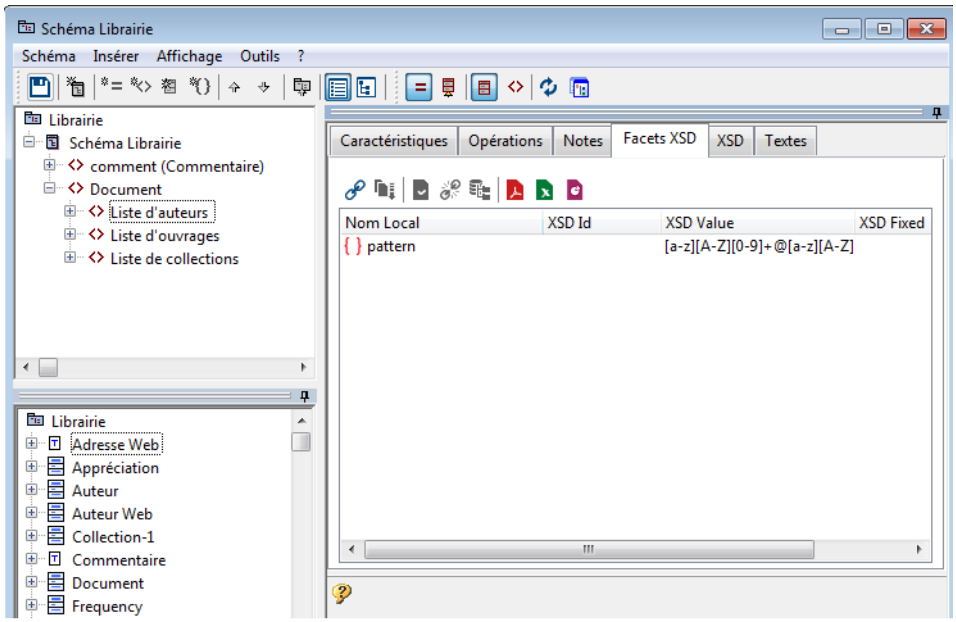
- **Final** : est modélisé par trois paramètres de génération représentant les types d 'héritage interdits :
- **XSD Final Restriction** à valeur tabulée "oui" ou "non"
- **XSD Final List** à valeur tabulée "oui" ou "non"
- **XSD Final Union** à valeur tabulée "oui" ou "non"
- **Id** : est modélisé par le paramètre de génération **XSD Id.**
- **Name** : est modélisé par le nom de la classe. Pour générer un nom différent de celui de la classe, entrez le nouveau nom dans le paramètre **XSD Name**.
- **Type** : il s'agit de modéliser quel est le type rétro-généré : type simple ou type complexe. Cet attribut est modélisé par le paramètre **XSD Type** qui prend les valeurs "type simple" ou"type complexe". Dans le cas présent, il prend la valeur "type simple". Il est facultatif.

Restriction

Le type de base ou le type interne de la restriction est modélisé par le lien de typage "UML type de base de l'expression" entre le type simple et le type de base. De plus, le type simple est de stéréotype "Expression". Si le type est local, on crée une classe imbriquée dans le type simple.

Les facets sont modélisées par des contraintes de stéréotype "XSD Facet" sur la classe (excepté pour Enumeration).

Dans l'exemple suivant, Adresse Web a une facet "pattern" qui a pour valeur '[a-z][A-Z][0-9]+@[a-z][A-Z][0-9]'.



Dans le cas d'une facet de type "énumération", le simpletype est de stéréotype "Enumeration". Les valeurs de l'énumération sont stockées en tant que "valeurs litérales".

Liste

La liste est modélisée de la manière suivante :

- Le type de base ou le type interne de la liste est modélisé par le lien de typage "UML type de base de l'expression" entre le type simple et le type de base. De plus, le type simple est de stéréotype "Expression". Si le type est local, on crée une classe imbriquée dans le type simple.
- "La liste est modélisée par le paramètre **XSD List** qui prend la valeur "oui" si s'agit d'une liste ou "non" dans le cas contraire.

Union

L'union est modélisée de la manière suivante :

- Les types peuvent être définis dans la liste de types memberTypes séparés par des espaces et/ou en local. Ils sont modélisés en tant qu'attributs de la classe associée au type simple. Le nom de l'attribut est le nom du type. S'il y a des types locaux, on crée des classes imbriquées dans le type simple.
- Le typage Union est modélisé par le paramètre **XSD Union** initialisé à "oui" ("non" dans les autres cas). Le type simple doit en plus être de stéréotype "Structure".

ComplexType

La balise "ComplexType" définit la structure d'une balise XML. Cette structure peut ensuite être réutilisée par plusieurs balises. La structure est complexe, c'est-à-dire qu'elle peut contenir des attributs et des sous-balises. On peut également hériter d'autres types en les restreignant (certains des éléments et attributs existants son restreints : ex : multiplicité) ou en ajoutant de nouveaux attributs et éléments.

La balise "ComplexType" est modélisée par une classe UML.

Attributs de "ComplexType"

- **Abstract** : est modélisé par l'attribut **Abstraite** de la classe UML associée au type.
- **Block** : est modélisé par les paramètres :
- **XSD Block Extension** : qui prend la valeur "oui" si on interdit la substitution du type par un type hérité par extension dans le document instance, la valeur "non" dans le cas contraire.
- **XSD Block Restriction** : qui prend la valeur "oui" si on interdit la substitution du type par un type hérité par restriction dans le document instance, la valeur "non" dans le cas contraire.
- **Final** : est modélisé par les paramètres :
- **XSD Final Extension** : qui prend la valeur "oui" si on interdit l'héritage par extension du complexType, la valeur "non" dans le cas contraire.
- **XSD Final Restriction** : qui prend la valeur 'oui' si on interdit l'héritage par restriction du complexType, la valeur "non" dans le cas contraire.
- **Id** : est modélisé par le paramètre **XSD ID**.
- **Mixed** : est modélisé par le paramètre **XDD Text**. Si la valeur de l'attribut vaut "yes", le paramètre XDD Text prend la valeur "oui". Dans le cas contraire, il prend la valeur "non".
- **Name** : définit le nom du type. Pour générer un nom différent de celui de la classe, indiquez le nouveau nom dans le paramètre **XSD Name**.
- **Type** : il s'agit de modéliser quel est le type rétro-généré : type simple ou type complexe. Cet attribut est modélisé par le paramètre **XSD Type** qui prend les valeurs "type simple" ou "type complexe". Dans le cas présent, il prend la valeur "type complexe". Il est facultatif.

SimpleContent

La modélisation dépend du type d'héritage :

Si le type restreint un type de base, on crée :

- Une généralisation entre les deux types. On initialise alors l'attribut de généralisation **XSD Heritated By Restriction** à "oui".
- Si la restriction porte des facets, on crée de la même façon que pour les types simples, des contraintes et on leur associe les valeurs des facets.
- Si la restriction restreint en plus des attributs du type de base, on crée l'attribut surchargeant, on relie l'attribut surchargeant à l'attribut surchargé par la patte "attribut surchargé".
- Si la restriction restreint un groupe d'attributs du type de base, on crée le groupe d'attributs surchargeant, on relie le groupe surchargeant au groupe surchargé par la patte "rôle surchargé".

Si le type étend un type de base, on crée :

- Une généralisation entre les deux types. On initialise alors l'attribut de généralisation **XSD Heritated By Restriction** à "non".
- On crée les nouveaux attributs et les nouveaux groupes.

ComplexContent

La modélisation dépend du type d'héritage :

Si le type restreint un type de base, on crée :

- Une généralisation entre les deux types. On initialise alors l'attribut de généralisation **XSD Heritated By Restriction** à "oui".
- Si la restriction restreint des éléments du type de base, on crée l'élément surchargeant, on relie l'élément surchargeant à l'élément surchargé par la patte "rôle surchargé".
- Si la restriction restreint des attributs du type de base, on crée l'attribut surchargeant, on relie l'attribut surchargeant à l'attribut surchargé par la patte "attribut surchargé".
- Si la restriction restreint un groupe d'attributs ou d'éléments du type de base, on crée le groupe surchargeant, on relie le groupe surchargeant au groupe surchargé par la patte "rôle surchargé".

Si le type étend un type de base, on crée :

- Une généralisation entre les deux types. On initialise alors l'attribut de généralisation **XSD Heritated By Restriction** à "non".
- On crée les nouveaux attributs, les nouveaux éléments et les nouveaux groupes.

Element

La balise "Element" est une déclaration d'un type simple ou d'un type complexe. C'est elle qui représente la balise instance du document. Elle peut contenir des attributs, des sous-balises et/ou des textes.

Un élément indique souvent une relation entre deux classes. Cette relation est représentée par une une composition UML. L'élément est modélisé par le rôle de la classe situé du côté de la classe associée à son type.

Attributs de "Element"

- **Abstract** : est modélisé par le paramètre de génération "XSD Abstract". Si l'élément est abstrait, sélectionnez "oui". Dans le cas contraire, sélectionnez "non".
- **Block** : est représenté par les trois paramètres de génération **XSD Block Extension**, **XSD Block Restriction** et **XSD Block Substitution**. Suivant la valeur de Block, les paramètres de génération prennent la valeur "oui" ou "non".
- **Default** : est modélisé par le paramètre de génération **XSD Value** qui contient la valeur par défaut de l'élément, et par l'attribut **IsChangeable** qui ne doit pas être spécifié.
- **Final** : est représenté par les paramètres de génération **XSD Final Extension** et **XSD Final Restriction**. Suivant la valeur de Final, les paramètres de génération prennent la valeur "oui" ou "non".
- **Fixed** : est modélisé par le paramètre de génération **XSD Value** qui contient la valeur par défaut de l'élément, et par l'attribut **IsChangeable** qui doit être initialisé à "frozen".
- **Form** : est représenté par le paramètre de génération **XSD Form**. Si l'élément est qualifié, le paramètre prend la valeur "Qualified". Dans le cas contraire, il prend la valeur "Unqualified".
- **MaxOccurs** : est représenté par la multiplicité supérieure du rôle de classe.
- **MinOccurs** : est représenté par la multiplicité inférieure du rôle de classe.
- **Name** : est modélisé par le nom du rôle.
- **Nillable** : est modélisé par le paramètre de génération **XSD Nillable**. Si l'attribut est nul, la valeur du paramètre est "oui". Dans le cas contraire, elle est à "non".
- **Ref** : est modélisé par le lien de référencement entre rôles de classe. Il s'agit du lien "Referenced Role/Referencing Role".
- **SubstitutionGroup** : est modélisé par le lien **rôle surchargé**. Ce lien est déjà utilisé pour la surcharge des éléments par restriction. L'attribut de lien **IsSubstituableTo** vaut "oui" si l'élément substitue un autre élément et vaut "non" si l'élément surcharge un autre élément.
- **Type** : est modélisé par la classe auquel le rôle de classe est associé.

Fils

ComplexType

Il s'agit d'un complexType interne à l'élément. Il est modélisé par une classe. Voir "ComplexType", page 143.

La définition locale du type est représentée par le lien "UML Classe Imbriquée". La classe est donc contenue par la classe associée.

SimpleType

Il s'agit d'un simpleType interne à l'élément. Il est modélisé par une classe. Voir "SimpleType", page 141.

La définition locale du type est représentée par le lien "UML Classe Imbriquée". La classe est donc contenue par la classe associée.

Unique - Key - KeyRef

Unique

La balise "Unique" définit une contrainte d'unicité sur un élément. La contrainte d'unicité s 'applique en général à la valeur d'un attribut ou au texte contenu dans une balise.

Unique : est modélisée par une contrainte de stéréotype 'XSD Uniqueness' et par un type de contrainte qui vaut 'unicité'. Cette contrainte est reliée au rôle de classe (élément) à contraindre.

Attributs de "Unique"

- **Id** : cet attribut n'est pas modélisé.
- **Name** : est modélisé par le nom de la contrainte.

Selector- Field

La balise "Selector" définit le chemin qui permet d'accéder aux champs contraints.

"Field" définit les attributs ou les éléments contraints. On accède à ces attributs ou éléments en parcourant le chemin défini par "Selector".

Ces balises sont modélisées par la zone texte **XSD Constraint** de la contrainte.

Annotation

Se reporter au paragraphe "Annotation (Documentation - AppInfo)", page 153.

Définition Key

La balise "Key" définit une contrainte d'unicité sur un élément. Elle a le même rôle que "Unique" mais vous pouvez en plus la référencer sous forme de clé étrangère. La contrainte d'unicité s'applique en général à la valeur d'un attribut ou au texte contenu dans une balise.

"Key" est modélisée par une contrainte de stéréotype "XSD Key". La contrainte n'a pas de type associé. Cette contrainte est reliée au rôle de classe (élément) à contraindre.

Attributs de "Key"

- **Id** : cet attribut n'est pas modélisé.
- **Name** : est modélisé par le nom de la contrainte.

Selector - Field

Ces balises sont modélisées par la zone texte "XSD Constraint" de la contrainte.

Définition KeyRef

La balise "KeyRef" définit une contrainte sur un élément qui doit prendre ses valeurs à partir des éléments soumis à une autre contrainte de type "Key".

"KeyRef" est modélisée par une contrainte de stéréotype "XSD KeyReference". La contrainte n'a pas de type associé. Cette contrainte est reliée au rôle de classe (élément) à contraindre.

Attributs de "Keyref"

- **Id** : cet attribut n'est pas modélisé.
- **Name** : est modélisé par le nom de la contrainte.
- **Refer** : est modélisé par le lien "Contrainte" entre la contrainte référençante et la contrainte référencée.

Selector- Field

Ces balises sont modélisées par la zone texte "XSD Constraint" de la contrainte.

Group

La balise "Group" définit un groupe d'éléments au niveau du schéma. Le but est de pouvoir réutiliser ce groupe dans différents schémas par le système de référencement. Ceci évite les redondances de définition de groupe d'éléments.

Le groupe est défini par une classe de stéréotype "schema Group". La déclaration du groupe est représentée par une composition UML. L'occurrence de groupe est modélisée par le rôle de classe du côté de la classe de stéréotype "schema Group". Le fonctionnement est le même que pour l'élément.

Attributs de "Group"

- **Name** : est modélisé par le nom du rôle de la classe et le nom de la classe : ils portent tous les deux le même nom.
- **ID** : est modélisé par le paramètre de génération **XSD ID** de la classe si le groupe est défini au niveau du schéma. Par contre, s'il s'agit d'une référence, on ne peut surcharger l'attribut Id de la classe. Il est alors modélisé par le paramètre de génération **XSD ID** défini dans l'onglet de génération du rôle de classe.
- **MaxOccurs** : est modélisé par la multiplicité maximale sur le rôle de classe.
- **MinOccurs** : est modélisé par la multiplicité minimale sur le rôle de classe.
- **Ref** : est modélisé par le lien de référencement entre rôles de classe. Il s'agit du lien "Referenced Role/Referencing Role".

Fils

Un groupe peut avoir comme fils :

- "All", page 148
- "Sequence", page 148
- "Choice", page 149
- "Annotation (Documentation - AppInfo)", page 153

All

La balise "All" définit un groupe d'éléments non ordonnés qui peuvent apparaître 0 ou une fois.

"All" est modélisée par une classe de stéréotype "schema Group" dans sa définition. La déclaration de ce groupe est représentée par une composition UML. Et l'occurrence de groupe est représentée par le rôle de classe UML du côté de la classe Groupe.

Le paramètre de génération **XDD Order** sur la classe vaut "aucun".

Si le groupe n'a pas de frères, le nom de la classe est "All". Sinon, il vaut "All"suivi de l'index d'ordre du groupe.

Si All n'a pas de frère, si sa multiplicité par défaut est 1, et si son père est un type ou un groupe modèle, le groupe All est factorisé dans le père. C'est le père qui porte le paramètre **XDD Order**.

Attributs de "All"

- **Id** : est modélisé par le paramètre de génération **XSD Id** sur la classe.
- **MaxOccurs** : est modélisé par la mutiplicité maximale sur le rôle de classe du groupe.
- **MinOccurs** : est modélisé par la mutiplicité minimale sur le rôle de classe du groupe

Fils

"All" peut avoir comme fils :

- "Annotation (Documentation - AppInfo)", page 153
- "Element", page 144

Sequence

La balise "Sequence" définit un groupe d'éléments et/ou de sous-groupes ordonnés de manière séquentielle.

"Sequence" est modélisée par une classe de stéréotype "schema Group" dans sa définition. La déclaration de ce groupe est représentée par une composition UML, et l'occurrence de groupe est représentée par le rôle de classe UML du côté de la classe Groupe.

Le paramètre de génération **XDD Order** sur la classe vaut "sequence".

Si le groupe n'a pas de frère, le nom de la classe est "Sequence". Sinon, il vaut "Sequence" suivi de l'index d'ordre du groupe.

Si Sequence n'a pas de frère, si sa multiplicité par défaut est 1 et si son père est un type ou un groupe modèle, le groupe Sequence est factorisé dans le père. C'est le père qui porte le paramètre **XDD Order**.

Attributs de "Sequence"

- **Id** : est modélisé par le paramètre de génération **XSD Id** sur la classe.
- **MaxOccurs** : est modélisé par la mutiplicité maximale sur le rôle de classe du groupe.
- **MinOccurs** : est modélisé par la mutiplicité minimale sur le rôle de classe du groupe.

Fils

"Sequence" peut avoir comme fils :

- "Element", page 144
- "Group", page 147
- "Choice", page 149
- "Sequence", page 148
- "Any", page 150
- "Annotation (Documentation - AppInfo)", page 153

Choice

La balise "Choice" définit un groupe d'éléments et/ou de sous-groupes. Dans le document instance, un seul de ces éléments ou groupes doit apparaître.

"Choice "est modélisée par une classe de stéréotype "schema Group" dans sa définition. La déclaration de ce groupe est représentée par une composition UML, et l'occurrence de groupe est représentée par le rôle de classe UML du côté de la classe Groupe.

Le paramètre de génération **XDD Order** sur classe vaut "choice".

Si le groupe n'a pas de frère, le nom de la classe est "Choice". Sinon, il vaut "Choice" suivi de l'index d'ordre du groupe.

Si Choice n'a pas de frère, si sa multiplicité par défaut est 1, si son père est un type ou un groupe modèle, le groupe Choice est factorisé dans le père. C'est le père qui porte le paramètre **XDD Order**.

Attributs de "Choice"

- **Id** : est modélisé par le paramètre de génération XSD Id sur la classe.
- **MaxOccurs** : est modélisé par la mutiplicité maximale sur le rôle de classe du groupe.
- **MinOccurs** : est modélisé par la mutiplicité minimale sur le rôle de classe du groupe.

Any

La balise "Any" définit un groupe qui peut contenir n'importe quel type d'élément. Vous pouvez cependant restreindre le groupe d'éléments à un ensemble d'espaces de nommage.

"Any" est modélisée par une classe de stéréotype "schema Group" dans sa définition. La déclaration de ce groupe est représentée par une composition UML, et l'occurrence de groupe est représentée par le rôle de classe UML du côté de la classe Groupe.

Le paramètre de génération **XDD Order** sur la classe vaut "aucun". Le paramètre de génération **XSD Any** vaut "any".

Si le groupe n'a pas de frère, le nom de la classe est "Any". Sinon, il vaut "Any" suivi de l'index d'ordre du groupe.

Attributs de "Any"

- **Id** : est modélisé par le paramètre de génération **XSD Id** sur la classe.
- **MaxOccurs** : est modélisé par la mutiplicité maximale sur le rôle de classe du groupe.
- **MinOccurs** : est modélisé par la mutiplicité minimale sur le rôle de classe du groupe.
- **Namespace** : est modélisé par le paramètre de génération **XSD Namespace.** Il est à valeurs tabulées et prend ses valeurs dans la liste suivante :
  - ##any : n'importe quel élément convient
  - ##other : n'importe quel élément n'appartenant pas à l'espace de nommage courant
  - ##targetNamespace : n'importe quel élément appartenant à l'espace de nommage courant
  - Liste d'uris : n'importe quel attribut appartenant à la liste des espaces de nommage
  - ##Local : n'importe quel élément local à un type
- **processContents** : est modélisé par le paramètre de génération **XSD processContents**. Il est à valeurs tabulées et prend ses valeurs dans la liste suivante :
  - lax : si l'élément a un type défini et connu, l'élément doit être valide par rapport à sa définition. Sinon, il doit juste être bien formé.
  - skip : aucune contrainte n'est appliquée. L'élément doit juste être bien formé.
  - strict : l'élément doit être validé à partir du moment où son type est idéntifié dans le document instance par un type défini ou par un type référencé par l'attribut xsi :type.

Attribute

La balise "Attribute" définit un attribut d'une balise XML. Elle est fondée sur un type simple.

L'attribut est modélisé par un attribut UML sur la classe qui le déclare. Ainsi, si l'attribut est défini au niveau du schéma, la classe qui porte l'attribut est la classe Schéma. Si on a affaire à un type, l'attribut sera sur la classe du type.

Attributs de "Attribute"

- **Default** : est modélisé par les attributs **Valeur initiale** et **Modifiable**. **Valeur Initiale** affiche une valeur par défaut et **Modifiable** vaut "oui".
- **Fixed** : est modélisé par les attributs **Valeur initiale** et **Modifiable**. **Valeur Initiale** affiche la valeur de Fixed et **Modifiable** vaut "non".
- **Form** : est modélisé par le paramètre de génération **XSD Form**. Cet attribut est à valeur tabulée et vaut "qualifié" ou "non qualifié".
- **Id** : est modélisé par le paramètre de génération **XSD Id**.
- **Name** : définit le nom de l'attribut. On définit également l'attribut **XSD Name** qui est la valeur prise en compte lors de la génération. Vous

pouvez donc modifier le nom de l'attribut sans changer la représentation du schéma.
- **Ref** : est modélisé par le lien "Referenced Attribute/Referencing Attribute". On référence un attribut défini au niveau du schéma.
- **Type** : est modélisé par le type de l'attribut.
- **Use** : est modélisé par la multiplicité de l'attribut.
  - Optional : la multiplicité de l'attribut vaut 0..1.
  - Required: la multiplicité de l'attribut vaut 1.
  - Prohibited: la multiplicité de l'attribut vaut 0

AttributeGroup

La balise "AttributeGroup" définit un regroupement d'attributs. Elle est déclarée au niveau du schéma puis référencée au sein des types. Cela évite la redondance de définition.

Le groupe d'attribut est défini par une classe de stéréotype "schema Group". La déclaration de ce groupe est représentée par une composition UML. L'occurrence de groupe est modélisée par le rôle de classe du côté de la classe de stéréotype "schema Group".

Attributs de "AttributeGroup"

- **Name** : est modélisé par le nom du rôle de la classe et le nom de la classe : ils portent tous les deux le même nom.
- **ID** : est modélisé par le paramètre de génération **XSD ID** de la classe si le groupe est défini au niveau du schéma. S'il s'agit d'une référence, vous ne pouvez surcharger l'attribut Id de la classe, il est alors modélisé par le paramètre de génération **XSD ID** défini dans l'onglet de génération du rôle de classe.
- **Ref** : est modélisé par le lien de référencement entre rôles de classe. Il s'agit du lien "Referenced Role/Referencing Role".

AnyAttribute

La balise "AnyAttribute" définit un groupe qui peut contenir n'importe quel type d'attribut. Vous pouvez cependant restreindre le groupe d'attributs à un ensemble d'attributs appartenant à certains espaces de nommage.

"AnyAttribute" est modélisée par une classe de stéréotype "schema Group" dans sa définition. La déclaration de ce groupe est représentée par une composition UML, et l'occurrence de groupe est représentée par le rôle de classe UML du côté de la classe Groupe.

Le paramètre de génération **XDD Order** sur classe vaut "aucun".

Le paramètre de génération **XSD Any** vaut "anyAttribute".

Si le groupe n'a pas de frère, le nom de la classe est "AnyAttribute" . Sinon, il vaut "AnyAttribute" suivi de l'index d'ordre du groupe.

Attributs de AnyAttribute

- **Id** : est modélisé par le paramètre de génération **XSD Id** sur la classe.
- **Namespace** : est modélisé par le paramètre de génération **XSD Namespace**. Il est à valeurs tabulées et prend ses valeurs dans la liste suivante :
  - ##any : n'importe quel attribut convient,
  - ##other : n'importe quel attribut n'appartenant pas à l'espace de nommage courant,
  - Liste d'uris : n'importe quel attribut appartenant à la liste des espaces de nommage,
  - ##targetNamespace : n'importe quel attribut appartenant à l'espace de nommage courant,
  - ##Local : n'importe quel élément local à un type.
- **processContents** : est modélisé par le paramètre de génération **XSD processContents**. Il est à valeurs tabulées et prend ses valeurs dans la liste suivante :
  - lax : si l'élément a un type défini et connu, l'élément doit être valide par rapport à sa définition. Sinon, il doit juste être bien formé.
  - skip : aucune contrainte n'est appliquée. L'élément doit juste être bien formé.
  - strict : l'élément doit être validé à partir du moment où son type est idéntifié dans le document instance par un type défini ou par un type référencé par l'attribut xsi :type.

Annotation (Documentation - AppInfo)

La balise "Annotation" permet d'associer une description à un concept XSD. La balise "Documentation" donne des informations humaines sur le concept. La balise AppInfo" fournit une information technique.

On distingue deux cas de modélisation :

Une seule annotation sur le concept

Si vous avez tout au plus une documentation et une appInfo sans source associée, la balise Documentation est modélisée par l'attribut **Comment** présent sur tous les concepts utilisés. La balise AppInfo est modélisée par le texte **XSD AppInfo** dans l'onglet génération XSD.

Plusieurs annotations ou plusieurs documentations et appInfos ou des sources associées

Dans ce cas, les balises Documentation et AppInfo sont modélisées par une Note. Pour la balise Documentation, la note est de stéréotype "XML Documentation". Pour la balise AppInfo, la note est de stéréotype "XML Application".

Attributs

- **Id** : n'est pas modélisé.
- **Source** : est modélisé par une référence externe associée à la note représentant "Documentation" ou "AppInfo".