Suivi d'édition

date	auteur	version	modif
16/09/02	TIF	1.0	• création par reprise du document d'analyse Configuration_v2_1.4.doc
30/09/02	TIF	1.0	• corrections
03/10/02	TIF	1.0	• corrections dans le paragraphe "Publications sur une ViewStruct".

Le but de ce document est de spécifier le paramétrage d'e-GX v4.4.0 et la manière de manipuler la configuration.

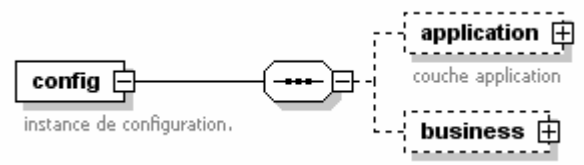
Structure générale

Le fichier de configuration est un fichier XML.

Ce fichier de configuration cumule principalement 2 rôles :

- technique : faire le lien entre l'instance d'e-GX et l'environnement cible (Base de données, répertoires, limites mémoires, …)
- fonctionnel : définition des modèles de données permettant l'execution des scénarii. (équivallent d'un dictionnaire de données).

Ces 2 rôles sont distincts des couches logicielles : application et métier. En effet, on prévoit à terme de déployer la couche application sur une machine différente de la couche métier.



Le premier découpage du fichier de configuration se représente par la séparation de la couche application de la couche métier. Puis, dans chaque couche, on peut faire la distinction entre les parties techniques et les parties fonctionnelles.

Couche Application

La partie "application" comporte onze sous-parties.

techniques	fonctionnelles
• properties • link_to_business • link_to_presentation	• documentrow_def • document_def • view_def • viewstruct_def • viewstruct_link_def • businessview_def • xsl_param • action_def

Paramètres techniques

Propriétés techniques de la couche Application : properties

Ce chapitre décrit le contenu du tag "properties".

```
<properties cacheActive="true" connection="soc1/infor1@oracle:1521:gnx" frameMaxNum="10" traceLevel="0"
xmlPresentationFile="debug_xml_presentation" uuidManage="true" defaultUserRecognize="true">
        <public_businessview>
            <businessview>VM1</businessview>
            <businessview>VM2</businessview>
            <businessview>VM3</businessview>
        </publicBusinessview>
        <visible bc4jViewLink="true" field="true"/>
    </properties>
```

connection : paramètre de connexion base de données du socle technique

Ce paramètre est optionnel en v4.4.0, il est présent pour de prochaines évolutions (connexions BDD spécifique à la couche application pour les données de configuration).

cacheActive : cache d'actions et de documents

"true" signifie que tout objet chargé depuis le système de fichier est conservé en mémoire pour accélerer les temps de chargement. Cela a pour conséquence que les modifications apportées à des fichiers ne seront visibles qu'au prochain redémarrage d'e-GX.

La valeur "false" ne se justifie que pour une problématique de développement ou les fichiers évoluent.

Les objets qui sont cachés :

- les feuilles de styles.
- les actions.
- les resolvers de clause retrieve ou de requête SQL dynamique.
- les documents.
- les définitions de ViewObject et ApplicationModule (BC4J).

```
exemple: cacheActive=true
```

traceLevel : niveau de trace

Permet d'appliquer un filtre sur les messages générés par le socle dans le fichier de log.

- 0 = Traces d'exploitation courante
- 1 = Traces de debuggage

```
traceLevel=0
```

frameMaxNum : Nombre maximum de frames

Nombre de frames maximum que le socle est autorisé à mémoriser, par session http.

```
exemple : frameMaxNum=10
```

workingDirectory : répertoire de travail du socle technique

Ce répertoire contient les fichiers générés par le socle (fichier du flux de présentation, fichier de log).

```
exemple : workingDirectory=log
```

xmlPresentationFile : XML de présentation

Nom du fichier dans lequel sera sauvegardé le flux XML de présentation retourné par une cinématique.

Ce paramètre est optionnel et n'est interprété que dans le cas `traceLevel=1`.

```
xmlPresentationFile = debug_xml_presentation.xml
```

defaultLoginBusinessview : le nom de la BusinessView de login

Cet attribut définit la BusinessView de login. C'est cette BV qui est utilisée pour gérer la cinématique de login.

Cette BusinessView est automatiquement publique.

```
exemple : defaultLoginBusinessview=LOGIN
```

uuidManage : activation ou désactivation de la gestion des UUID

La valeur par défaut est "true".

Si "true", alors on gére le UUID par session. On reconnaît un utilisateur qui dispose d'un cookie Générix. Tout poste client qui n'a pas le cookie générix se verra doté d'un nouveau uuid personnel. Cela permet de gérer une session logique (concept permettant de suivre différement chaque utilisateur non logué, donc "GUEST").

Si "false", désactive la gestion de la session logique. Cela ne supprime pas les cookies sur le poste client. Les cookies sont justes ignorés.

UUID = Unique Universal Identifier.

defaultUserRecognize : login automatique

La valeur par défaut est "true". Cet attribut n'a aucun sens si uuidManage est à "false".

Si "true", alors le support de présentation est reconnecté pour l'utilisateur précédement logué. Grâce à l'UUID stocké dans le cookie du navigateur, on reconnaît l'utilisateur précédement logué et on le réidentifie automatiquement.

Si "false", alors on considère que toutes les sessions logiques sont associées à l'utilisateur GUEST.

checkId : vérification de la contrainte de continuité.

Dans chaque flux de présentation, il y a un attribut id qui est généré. Cet attribut dépend directement de l'état de la session http au moment de la génération du xml de présentation. Lorsque cette option est activée, elle permet de vérifier qu'on ne tente pas d'appliquer une url qui n'est pas continue par rapport à l'état du serveur.

Un identifiant de requête est ajouté au XML de présentation afin de garantir l'intégrité entre les états courants sur le serveur et ce qui est présenté à l'utilisateur.

Le XML de présentation contient une valeur correspondant à un checksum représentant l'état de l'application lors de la génération de la présentation.

Toutes les urls doivent renvoyer cet identifiant pour assurer la continuité du traitement entre deux urls.

L'identifiant se trouve à l'emplacement suivant dans le XML de présentation :
`/layout_data/application_data/id`

L'identifiant doit être renvoyé sous la forme d'un attribut nommé "id".
par exemple : `id=531b02df`

En cas de différence entre l'identifiant renvoyé sur l'url et l'identifiant attendu par le serveur, la requête courante est perdue et est remplacée automatiquement par la requête suivante : cinematic=display(0) et une erreur est ajoutée dans le XML de présentation.

Quand ce cas d'erreur se produit, cela signifie que l'utilisateur visualise une BusinessView qui n'est plus réellement la courante ou la même dans un état qui n'est plus celui du serveur. L'effet du display(0) est d'afficher à l'utilisateur l'état réel du contexte pour le serveur dans le but de reprendre une cinématique valide.

L'utilisation de l'attribut "sourceview" sur l'url implique que l'attribut "id" n'est pas obligatoire.

Le tag représentant l'erreur se trouve dans le flux de présentation à l'emplacement suivant :
`/layout_data/error`

Liste des BusinessView publiques : tag "publicBusinessView"

C'est une façon rudimentaire de gérer les droits d'accès. On définit une liste des BusinessView publiques. Tant qu'un utilisateur est 'GUEST', il ne peut afficher que des BV publiques. S'il demande une cinématique sur une BV privée (absente de la liste), alors une page de login lui demandera de s'identifier avant de lui donner accès à l'affichage demandé.

Ce fonctionnement est validé par le responsable produit.

Visibilité des champs et des ViewLink Oracle : tag "visible"

Définit la visibilité par défaut des champs simples et des champs de type ViewLink.

- field : champs simples.
- bc4jViewLink : ViewLink Oracle.

Des niveaux plus fins de paramétrage sont disponibles dans les ViewStruct.

Lien avec la couche métier : link_to_business

Le lien entre la couche Application et la couche Métier s'effectue par l'Application Module "SessionMetier". SessionMetier est un ApplicationModule technique de la couche Métier.

Configurer les liens entre la couche Application et la couche Métier revient à définir comment "contacter" une instance de SessionMetier. Pour cela, on définit un contexte JNDI et des paramètres spécifiques à un ApplicationModule root.

Contexte JNDI : jndi

Sous le nœud "link_to_business", on trouve le nœud "jndi" qui porte les attributs suivants :

attribut	description
deploy_platform	plateforme de déploiement () : deux valeurs possibles ("local" ou "ejb"). Actuellement, seul le paramètre "local" est utilisable.
application_path	chemin du contexte courant
principal	utilisateur
credentials	mot de passe
authentication	type d'authentification. deux valeurs possibles ("non_ssl_login" ou " ssl_login")
port	numéro de port
host	machine hôte
sid	nom d'instance de base Oracle

Cela permet de définir les paramètres contextuels liés à la connexion entre la couche technique application et la couche technique métier.

SessionMetier : business_session

Nous considérons qu'il existe deux types d'ApplicationModule du point de vue du socle technique : les am "root" et les am "nested".

En effet pour chacun, de ces deux types d'am, nous ne définirons pas les mêmes caractéristiques. Un am "nested" comprend un sous-ensemble de paramètres de l'am "root".

	nested LOCAL	root LOCAL
stateless/statefull	oui	oui
chaîne de connexion	non	oui
verrouillage	non	oui
DefFullName	oui	oui

gestion d'instance (stateless ou statefull) : ce paramètre indique au socle technique, qui gère les instances d'ApplicationModule "nested", quelle est la politique de gestion des ApplicationModule. Dans ce cas, le socle technique se substitue au serveur d'application pour gérer les instances de composants.
Nous recommandons de n'utiliser que "statefull".

chaîne de connexion : représente la chaîne de connexion JDBC permettant de se connecter à la base. Seuls les ApplicationModule root se connectent à la BDD.
Avec Oracle 9iAS, la chaîne de connexion de la configuration devient optionnelle. Dans ce cas, cette donnée se trouve dans les data-sources du serveur d'application.

```
exemple: connection=jdbc:oracle:thin:soc1/infor1@oracle:1521:gnx
```

verrouillage (optimistic ou pessimistic) : le niveau de verrouillage se paramètre sur la transaction. Comme seul l'ApplicationModule "root" a une connection sur la BDD, donc seul l'ApplicationModule "root" gère la transaction donc seul l'ApplicationModule "root" est concerné par le paramètre verrouillage.

Nombre d'instances dans le pools d'ApplicationModule root : minInstance définit le nombre d'instances pré-instanciées à la construction du pool. maxInstance définit le nombre maximal de sessions utilisateurs possibles sur

l'instance d'e-GX. Ces deux attributs sont optionnels. S'ils ne sont pas présents, des valeurs par défaut (minInstance=0, maxInstance=5) sont prises en compte.

exemple

```
<link_to_business>
    <jndi deployPlatform="local" authentication="non_ssl_login" port="2481" host="oracle" sid="gnx"
principal="scott" credentials="tiger"/>
    <business_session sessionType="statefull"
defFullName="fr.generix.metier.bc4j.parametrage.SessionMetier"
connection="jdbc:oracle:thin:scott/tiger@oracle:1521:gnx" lockingMode="pessimistic  minInstance="2"
maxInstance="20"/>
</link_to_business>
```

Lien avec la présentation : link_to_presentation

Certaines informations concernant la relation entre la couche de présentation et la couche application doivent être modifiables. Notamment les paramètres permettant de gérer l'identification automatique.

Présentation http

Le socle utilise un cookie afin de mémoriser un identifiant unique caractérisant un poste utilisateur. Le cookie se définit de la façon suivante :

- name : un nom technique interne au socle. **Il est interdit de modifier cette valeur.**
- cookieName : le nom du cookie (qui sera dans le cookie mais n'est pas le nom du fichier).
- maxAge : une durée de vie (en secondes).
- domain : représente le domaine d'application du cookie. Cela permet au cookie d'être lu par toutes les machines d'un même domaine, et pas seulement par celle qui est productrice du cookie. Les noms de domaines doivent commencer par un point (exemple .generix.fr). Par défaut, si le domaine n'est pas renseigné, seule la machine productrice a accès au cookie.
- path : spécifie un chemin à partir duquel le cookie peut être lu. Si ce chemin n'est pas précisé, le cookie sera lisible depuis le répertoire et ses sous-répertoires de la page qui l'a envoyé. Le répertoire d'envoi doit par ailleurs obligatoirement faire partie du chemin spécifié. Nous conseillons "/" comme valeur.

```
<link_to_presentation>
    <http>
        <cookie name="UUID" cookieName="UUID" maxAge="10000" path=""/>
    </http>
</link_to_presentation>
```

Optimisations

Fréquence d'appel de executeQuery()

L'exécution d'une requête SQL est coûteuse. Il est possible de paramétrer la fréquence de cette exécution selon deux modes :

- "always" : A chaque fois que le document est consulté. Ce le mode le plus sûr. Il est conseillé soit pour visualiser des données fréquemment mises à jour soit avant de faire une modification.
- "optimize" : A chaque fois que la clause "where" est modifiée. Ce mode est le plus performant. Il est conseillé pour visualiser des données qui varient peu.

C'est un paramètre de la couche application. Il est défini par l'attribut executeQuery.

Le paramétrage de cette option d'optimisation peut se faire à deux niveaux :

par vue métier : (pour chaque nœud businessview dans la configuration). Pour chaque vue métier, on peut spécifier une fréquence d'appel de executeQuery() sinon, c'est le paramètre global qui est pris en compte.

global : (sur le nœud businessview_def de la configuration). Fixe une valeur par défaut pour toutes les vues métier qui n'ont pas de valeur spécifique.

```
<businessview_def executeQuery="optimize">
    <businessview name="vueA" viewstruct="VueA" target="CLI" executeQuery="optimize"/>
    <businessview name="vueB" viewstruct="VueB" target="CLI" executeQuery="always"/>
    <businessview name="vueC" viewstruct="VueC" target="CLI"/>
</businessview_def>
```

Dans cet exemple, on a modifié le comportement par défaut de executeQuery et on a redéfini l'optimisation pour certaines vues métier.

Si le paramètre executeQuery n'est pas présent, le mode de travail par défaut est "optimize".

Paramètres fonctionnels

Définition des lignes de Document

Les lignes de Document paramétrés dans la configuration ne concernent que les Documents de type APIDocument.

La définition des lignes de documents permet de créer des références utilisables par la configuration pour désigner les lignes des APIDocument, donc les Documents qui travaillent avec les ApplicationModule.

Une ligne de Document est caractérisée par :

- un nom : le nom repris par les Document pour désigner cette ligne de Document.
- une classe : le nom (package inclus) du DocumentRow spécifique développé pour invoquer le service.
- un service : le nom du service invoqué sur l'ApplicationModule associé au Document.

```
<documentrow_def>
    <documentrow name="GestionEvenementcreerPosteRow" class="fr.generix.metier.document.GestionEvenementcreerPosteRow"
service="creerPoste"/>
    <documentrow name="GestionEvenementdupliquerEvenementRow"
class="fr.generix.metier.document.GestionEvenementdupliquerEvenementRow" service="dupliquerEvenement"/>
    <documentrow name="GestionEvenementrevaloriserRow" class="fr.generix.metier.document.GestionEvenementrevaloriserRow"
service="revaloriser"/>
</documentrow_def>
```

Définition des documents

La définition des documents permet de créer des références utilisables par la configuration pour désigner les Documents.

De plus, cela permet de modifier la définition d'un document sans impacter l'ensemble des références qui y seront faites par le reste de la configuration.

```
<document_def>
    <document name="DocTie" class="fr.generix.package.DocGenerique" viewobject="JTieView"/>
    <document name="DocEve" class="fr.generix.package.DocGenerique" viewobject="JEveView"/>
    <document name="DocPoste" class="fr.generix.package.DocSpecifique1" viewobject="JPosteView"/>
    <document name="DocService" class="fr.generix.package.DocSpecifique2" applicationmodule="Am1"/>
</document_def>
```

Les Documents se classent en deux catégories

Document travaillant avec une ViewObject

La configuration liée à un Document de ce type est caractérisé par :

Attribut	Fonction	Obligatoire
name	Nom auquel fait référence une View qui travaille avec ce Document	X
class	la classe Java correspondant à ce type de Document. Non modifiable.	X
viewobject	Nom Générix de la ViewObject avec laquelle travaille ce Document.	X

```
<document_def>
    <document name="DocUtilisateur" class="fr.generix.technicalframework.application.document.RowDocumentImpl"
viewobject="UtUtiView"/>
</document_def>
```

Document travaillant avec un ApplicationModule

La configuration liée à un Document de ce type est caractérisée par :

Attribut	Fonction	Obligatoire
name	Nom auquel fait référence une View qui travaille avec ce Document.	X
class	Classe Java correspondant à ce type de Document. Non modifiable.	X
documentrow	Nom de la ligne de Document piloté par ce Document.	X
applicationmodule	Nom Générix de l'ApplicationModule avec lequelle travaille ce Document.	X

un nom : c'est le nom auquel fait référence une View qui travaille avec ce Document
une classe : : c'est la classe Java correspondant à ce type de Document. Non modifiable.
un nom d'ApplicationModule : nom GénériX de l'ApplicationModule avec lequel va travailler ce Document.

```
<document_def>
    <document name="uuidManageDocument" documentrow="uuidManageRow"
class="fr.generix.technicalframework.application.document.APIDocumentImpl" applicationmodule="UuidManager"/>
</document_def>
```

Définition des types de vue

Un type de vue porte un nom et fait référence à un document.
De plus, un type de vue définit les clauses "retrieve" et "sort" qu'il est possible d'utiliser sur lui-même.

Configuration de la clause retrieve

La définition d'une clause retrieve dans la configuration diffère selon le type de document. Voici la définition et le rôle des attributs pour chaque type de document.

Dans le cas des RowDocument, il faut être capable de passer des paramètres.
Afin de définir les emplacements des variables dans les clauses retrieve, nous avons choisi un caractère inutilisé par la norme SQL.
Historiquement, le '@' a déjà été utilisé par le socle Geneat pour définir des variables dans les maquettes. Nous optons donc pour ce caractère.
On peut aussi utiliser le caractère '?' pour représenter une variable SQL. Ce caractère est reconnu par le parser SQL d'Oracle et donne de meilleures performances. Quand on utilise le '?', on est dispensé de se poser la question du type du paramètre.

Vue travaillant avec un RowDocument et une ViewObject standard

Le nœud définissant la clause "retrieve", pour une View travaillant avec un StdRowDocument, porte le nom **where_clause.**

Attribut	Rôle	Obligatoire
key	Nom de clé publique.	X
value	Chaîne de caractères représentant la clause where avec ses inconnus.	

resolverClass	Nom de la classe (package inclus) résolvant la clause retrieve	X

exemple :

```
<retrieve>
    <where_clause key="parClient" value="Noclient=@"
resolverClass="fr.generix.technicalframework.application.DefaultWhereClauseResolver"/>
</retrieve>
```

Vue travaillant avec un RowDocument et une ViewObject dynamique

Le nœud définissant la clause "retrieve", pour une View travaillant avec un DynRowDocument, porte le nom **sql_query**.

Attribut	Rôle	Obligatoire
key	Nom de clé publique.	X
value	Chaîne de caractères représentant la requête SQL avec ses inconnus.	
resolverClass	Nom de la classe (package inclus) résolvant la clause retrieve	X

exemple:

```
<retrieve>
    <sql_query key="parCommande" value="select Nocmde,Produit,Qte from Roj_Lg_Cmde where Nocmde=@"
resolverClass="fr.generix.technicalframework.application.DefaultSQLQueryResolver"/>
</retrieve>
```

Vue travaillant avec un APIDocument et un ApplicationModule

Le nœud définissant la clause "retrieve", pour une View travaillant avec un APIDocument, porte le nom **api_clause**

Attribut	Rôle	Obligatoire
key	Nom de clé publique.	X
service	Nom du service invoqué.	X
resolverClass	Nom de la classe (package inclus) résolvant la clause retrieve	X

exemple :

```
<retrieve>
    <api_clause key="key1" service="retrieve" resolverClass="fr.generix.technicalframework.application.DefaultAPIParamResolver"/>
</retrieve>
```

Configuration de la clause sort

La clause sort ne concerne que les vues travaillant avec un RowDocument et une ViewObject, qu'elle soit standard ou dynamique. Il n'y a pas de tri sur les APIDocument.

Attribut	Rôle	Obligatoire
key	Nom de clé publique.	X
value	Chaîne de caractères représentant la clause sort.	X

Le nœud définissant la clause sort porte le nom **orderby_clause**
exemple :

```
<sort>
    <orderby_clause key="desc" value="No desc"/>
</sort>
```

Nota : la notion de résolveur de clause sort n'existe pas dans la version 440.

Mise en garde sur l'écriture d'une clause retrieve

Etant donné que nous ne gérons pas de dictionnaire de données, nous ne connaissons ni les noms des champs de chaque vue, ni leur type. Or, pour écrire une clause retrieve, il faut tenir compte du type de la donnée afin de l'encadrer de quotes s'il s'agit d'une chaîne de caractères.

Il sera de la responsabilité de la personne qui écrira les clauses retrieve de vérifier les types des champs dans un système externe à e-GX, et d'en tenir compte afin de formater correctement la clause where.

Ceci est vrai pour les clauses "retrieve" dont les inconnus sont des '@'. Dans le cas de l'utilisation des '?', il faut juste placer le caractère '?' à l'emplacement de la valeur sans tenir compte du type.

exemple de configuration.

```
<view_def>
    <view_type name="VueClient" document="ClientDoc">
        <retrieve>
            <where_clause key="clientKey" value="nomClient=@ "
resolverClass="fr.generix.technicalframework.application.DefaultWhereClauseResolver"/>
        </retrieve>
        <sort>
            <orderby_clause key="desc" value="No desc"/>
        </sort>
    </view_type>
    <view_type name="VueCommande" document="CmdeDoc">
        <retrieve>
            <where_clause key="parClient" value="Noclient=@"
resolverClass="fr.generix.technicalframework.application.DefaultWhereClauseResolver"/>
        </retrieve>
        <sort>
            <orderby_clause key="desc" value="No desc"/>
        </sort>
    </view_type>
    <view_type name="VueLigneCommande" document="LgCmdeDoc">
        <retrieve>
            <sql_query key="parCommande" value="select Nocmde,Produit,Qte from LgCmde where Nocmde=@"
resolverClass="fr.generix.technicalframework.application.DefaultSQLQueryResolver"/>
        </retrieve>
        <sort>
            <orderby_clause key="desc" value="Nocmde desc"
        </sort>
    </view_type>
</view_def>
```

Définition des ViewStruct

Une **ViewStruct** est une publication de composite de vues.
Une ViewStruct représente un type de vue métier.
Une ViewStruct possède les attributs suivants :

- un nom de type.
- une clé retrieve par défaut.
- une clé sort par défaut.

Une ViewStruct référence une View principale de la ViewStruct.

Définition des instances de View

Une **View** est composite de View.
Une View possède les attributs suivants :

- un type qui est une référence sur un nom de type de vue.
- un "nom publié" (attribut "name") permettant aux actions de faire référence à la View à partir de l'URL (par exemple les actions de navigation).
- un nombre de lignes qui est le nombre de row que retournera le document de la vue quand on lui demandera des données. Dès qu'une instance de vue possède un nombre lignes strictement supérieur à 1, alors les champs de cette instance de vue sont des colonnes de tableau. Il y aura donc des indices de ligne à gérer.
- un attribut booléen qui autorise de passer le document de l'instance de vue en création.
- un nom de document partagé

Une instance de View se trouvant dans une seule ViewStruct, on peut définir les **noms des champs** qui seront **publiés** pour la ViewStruct.

Une instance de View possède toujours un élément parent :

- la ViewStruct elle-même pour la première View
- la View parent pour une View composite.

On peut définir des liens entre :

- une instance de View et son élément source direct.
- une instance de View et le ViewStruct dans lequel elle est instanciée.

Ce lien se nomme ViewLink et sera définit dans un paragraphe suivant.

exemple de configuration.

```
<view type="VueClient" nbline="3" name="vueClient" create="true" docSharedName="docClient">
```

Publications sur une ViewStruct

Sur une ViewStruct, nous allons publier plusieurs informations :

- des caractéristiques de champ
- des caractéristiques de ViewLink Oracle
- des noms de View
- des clés de "rel" pour les clauses where et orderby

Caractéristiques de champs

Les champs sont les éléments constitutifs des lignes d'une View. Le tag "**field**" est positionné sous le tag "**view**" dans la définition de la ViewStruct. Le tag "field" contient la liste des champs de la View, le tag "**attribute**".
Un champ correspond à un champ dans une view de la ViewStruct et peut avoir comme caractéristique :

- **Un nom de champ publié**

On peut définir des **champs publiés** pour chaque ViewStruct.
Un **champ publié** correspond à un champ dans une view de la ViewStruct.
Il y a une relation 1-1 entre un champ d'une instance de View et un champ publié de ViewStruct.
Conseil : Eviter de publier tous les champs d'un document s'il n'y en a que quelques uns d'utilisés sur l'url.

- **Champs visibles**

Chaque ViewStruct peut définir la visibilité de ses champs. Ce paramétrage s'applique seulement aux champs qui ne sont pas des ViewLink Oracle (pour ceux là, il y a un autre paramétrage. cf. "view_link/bc4j").
Un champ peut être défini comme visible ou invisible. Il y a plusieurs niveaux de paramétrage.
Par ordre décroissant de priorité :

- champ : tag "attribute"
- vue : tag "field"

- globalement à toute l'application : "/config/application/properties/visible"

- **Les attributs du nœud "attribute"**

Attribut	Rôle	Obligatoire
visible	Visibilité. Présent sur les nœuds field et attribute.	
name	Nom interne. Correspond au nom du champ de la ViewObject.	X
public	Nom de champ publié.	

exemple de configuration

```
<field visible="true">
    <attribute name="No" public="No" visible="false"/>
    <attribute name="Nom" public="NOM"/>
    <attribute name="Prenom" public="PRENOM"/>
</field>
```

Caractéristiques de ViewLink Oracle

Tout comme pour les champs "classiques", il est possible de définir la visibilité des ViewLink. Un ViewLink peut être défini comme visible ou invisible. Par ordre décroissant de priorité : ViewLink, View ou globalement à toute l'application.

Ce paramétrage vient se placer dans le paragraphe ViewLink, au même niveau que les nœuds "retrieve" et "sort". Les champs (tag "attribute") sont encadrés par le nœud nommé "bc4j".

- **Les attributs du nœud "attribute"**

Attribut	Rôle	Obligatoire
visible	Visibilité. Présent sur les nœuds bc4j et attribute.	
name	Nom interne. Correspond au nom du champ de type ViewLink Oracle de la ViewObject.	X

exemple de configuration

```
<bc4j visible="false">
    <attribute name="CodpostViewLink" visible="true"/>
</bc4j>
```

View publiées

Un **nom de View publié** correspond à une instance de View de la ViewStruct.

Ce nom permet aux actions de faire référence directement à une vue particulière dans la BusinessView courante afin d'appliquer un traitement à cette seule vue (par exemple les actions de navigation, de suppression).

Les noms de View publiées ne peuvent pas faire double emploi avec les noms de champs. Ces derniers conservent leur rôle de passage de paramètre entre les ViewStruct.

exemple de configuration

```
<view type="VueClient" name="vueClient">
```

clé de Rel publiées

Une **clé de "rel" publiée** permet d'associer, pour chaque ViewStruct, les clés de "rel" des instances de View (retrieve et sort). Cette clé (attribut **publicKey**) fait le lien entre l'utilisateur (la requête) et la définition de la clé dans la définition des types de vues (attribut **viewKey**).

Exemple de configuration

```
<retrieve publicKey="CLI" viewKey="clientKey">
```

ViewLink

Un viewlink dans la configuration permet d'initialiser l'instance de View.
Dans une instance de View, on peut initialiser quatre types d'éléments :

- valeurs des champs de l'instance de View
- définitions et publication des clauses retrieve
- publication des clauses sort
- paramétrage du XML de présentation des maîtres détail.

valeurs des champs de l'instance de View

C'est une affectation entre deux champs de la ViewStruct. Le champ destination doit exister dans l'instance de vue. Comme il n'y a pas de dictionnaire de données, c'est à la personne qui écrit les noms de champs destination d'une affectation de connaître les noms et les types des champs des View.

définitions et publication des clauses retrieve

Une instance de View peut définir plusieurs clés de rel publiques pour la clause retrieve.
Pour chaque clé de rel publique, on fait référence à une clé de rel définie par le type de View et on renseigne les paramètres permettant d'initialiser les variables de la clause retrieve.

Un paramètre peut avoir plusieurs sources :

- "business" : la valeur du champ correspond à une valeur métier dans la View parent.
- "input" : la valeur du champ correspond à une valeur saisie dans la View parent.
- "public" : la valeur du champ correspond à la valeur d'un champ publié sur la ViewStruct.
- "value" : la valeur du champ est la valeur à utiliser. C'est pour passer des valeurs en dur.

exemple de configuration
si la clause retrieve correspondant au rel '0' dans le type de la View est :

```
POSTE like '@%' AND AUTRE='@'
```

```
<view_link>
    <retrieve publicKey="A" viewKey="0">
        <param domain="business" field="poste"/>
        <param domain="value" field="3"/>
    </retrieve>
    <sort publicKey="0" viewKey="0"/>
</view_link>
```

publication des clauses sort

Une instance de View peut définir plusieurs clés de rel publiques pour la clause sort (indépendamment des clés publiques de clause retrieve).
De même que pour la clause retrieve, pour chaque clé de rel publique, on fait référence à une clé de rel définie par le type de View. La différence avec la clause retrieve est qu'il n'y a pas de paramètre pour une clause sort

exemple de configuration

```
<view_link verboseDetail="true">
```

```
        <sort publicKey="inverse" viewKey="desc"/>
    </view_link>
```

XML de présentation des maître/détail

Un attribut optionnel est disponible sur la définition d'un ViewLink. Cet attribut se nomme "verboseDetail" et peut prendre la valeur "true" ou "false".
Cet attribut définit la politique de génération du XML de présentation pour ce ViewLink.

Cet attribut est exploité si et seulement si la view est une View dépendante. Rappelons au passage qu'une View est dépendante si un de ses paramètres de clause retrieve fait parti du domaine "business".

verboseDetail="false" (valeur par défaut) et la View est dépendante : le XML de présentation ne contient que le XML correspondant au détail de l'élément courant de la View maître.

verboseDetail="true" et la View est dépendante : le XML de présentation contient le XML correspondant aux détails de toutes les lignes visibles de la View maître.

exemples :
avec verboseDetail="true"

```
<view>
   <viewrow>valeur_1
      <view>
         <viewrow>valeur_1.1</viewrow>
         <viewrow>valeur_1.2</viewrow>
         <viewrow>valeur_1.3</viewrow>
      </view>
   </viewrow>
   <viewrow current="true">valeur_2
      <view>
         <viewrow>valeur_2.1</viewrow>
         <viewrow>valeur_2.2</viewrow>
         <viewrow current="true">valeur_2.3</viewrow>
      </view>
   </viewrow>
   <viewrow>valeur_3
      <view>
         <viewrow>valeur_3.1</viewrow>
         <viewrow>valeur_3.2</viewrow>
         <viewrow>valeur_3.3</viewrow>
      </view>
   </viewrow>
</view>
```

avec verboseDetail="false"

```
<view>
   <viewrow>valeur_1   </viewrow>
   <viewrow current="true">valeur_2
      <view>
         <viewrow>valeur_2.1</viewrow>
         <viewrow>valeur_2.2</viewrow>
         <viewrow current="true">valeur_2.3</viewrow>
      </view>
   </viewrow>
   <viewrow>valeur_3</viewrow>
</view>
```

La saisie dans une View de détail mérite d'être développée. Si on saisie dans une Views de détail, c'est qu'elle possède au moins un champ public. Rappelons que le moyen de rendre une View non saisissable est de ne pas définir de champs public sur cette dernière.

Si l'attribut "verboseDetail" vaut "true", dès qu'il y a une saisie sur cette View, cet attribut est ignoré et le XML de présentation est généré comme si "verboseDetail" était positionné à "false".

Ceci est dû à un compromis pour les performances d'e-GX. Vouloir faire une saisie dans une View de détail et continuer à afficher le XML de présentation pour toutes les lignes de la View maître est structurellement impossible sans des pertes de performances inacceptables.

Résumé :

Sur une View de détail, si on désire afficher tous les détails de toutes les lignes de la View maître **et** faire de la saisie, il faut savoir que dès qu'il y aura un propose(), il n'y aura plus que le détail de l'élément courant du maître qui sera affiché et la saisie aura lieu dans cette collection de lignes.
Puis, après un assimilate() (quand il n'y aura plus de saisie en cours) la view reprendra son comportement initial et affichera tous les détails de toutes les lignes de la View maître.

Sur une View de détail, si on désire **toujours** afficher tous les détail de toutes les lignes de la View maîtres, il faut définir l'attribut "verboseDetail" à "true" **et** ne pas définir de champs publics pour cette View. Donc, cela se "paye" par l'impossibilité de faire une saisie.

Partage de documents

Quand on définit un nom de partage de document, on crée une référence globale à la configuration.

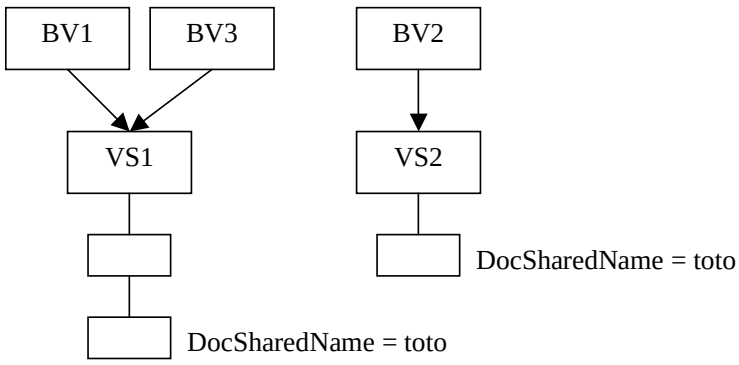
Toute View qui utilise le même nom de document partagé, partagera son instance de document avec les View qui utiliserons le même nom.

Si des View, n'utilisant pas le même type de document, définissent le même nom de partage, cela sera une erreur de configuration.

remarques importantes

- Il n'y a pas de partage de document entre des documents issus de la même ViewStruct.
- Il est interdit d'utiliser le partage de document sur des APIDocument.
  Cette limite est liée au cahier des charges v4.4.0. Le partage d'APIDocument provoquera le plantage de l'application.

Remarquons que le cas suivant peut se produire.
Soit la configuration suivante :



Quand on navigue de la BusinessView BV1 vers la BV2, puis sur BV3, c'est le même document qui est partagé à chaque navigation car on change de BusinessView à chaque navigation.
Alors que, dans une navigation qui va de la BusinessView BV1 vers la BV3 : on aura une nouvelle instance de document.

Dans ce cas, le partage de document n'a pas le même comportement suivant la cinématique utilisée pour aller d'un document à un autre.

Finalement, pour partager un document entre des ViewStruct semblables, on dupliquera les ViewStruct avec des noms différents.

Exemple de configuration.

```
<view type="VueClient" nbline="3" name="vueClient" docSharedName="docClient">
```

Conclusion

La définition d'une ViewStruct fait entrer un grand nombre de paramètres. C'est une partie importante de la configuration de la couche Application.

Voici pour terminer un exemple de configuration d'une ViewStruct complète.

```
<viewstruct name="VSCommandesParClient" defaultRetrieve="CLI" defaultSort="inverse">
    <view type="VueClient" nbline="3" name="client" create="true">
        <view_link verboseDetail="true">
            <retrieve publicKey="CLI" viewKey="clientKey">
                <param domain="public" field="NO"/>
            </retrieve>
            <sort publicKey="inverse" viewKey="desc"/>
        </view_link>
        <field>
            <attribute name="No" public="NO"/>
            <attribute name="Nom" public="NOM"/>
            <attribute name="Prenom" public="PRENOM"/>
            <attribute name="Age" public="AGE"/>
        </field>
        <view type="VueCommande" nbline="3" name="commande" create="true">
            <view_link verboseDetail="false">
                <retrieve publicKey="CLI" viewKey="parClient">
                    <param domain="business" field="No"/>
                </retrieve>
                <sort publicKey="inverse" viewKey="desc"/>
            </view_link>
            <field>
                <attribute name="Datecmde" public="DATECMDE" visible="true"/>
                <attribute name="No" public="NO" visible="true"/>
                <attribute name="Noclient" public="NOCLIENT" visible="false"/>
            </field>
            <view type="VueLigneCommande" nbline="3" name="ligneCommande" create="true">
                <view_link verboseDetail="true">
                    <retrieve publicKey="CLI" viewKey="parCommande">
                        <param domain="business" field="No"/>
                    </retrieve>
                    <sort publicKey="inverse" viewKey="desc"/>
                </view_link>
                <field>
                    <attribute name="Qte" public="QUANTITE" visible="true"/>
                    <attribute name="Nocmde" public="NOCMDE" visible="false"/>
                    <attribute name="Produit" public="PRODUIT" visible="true"/>
                </field>
            </view>
    </view>
</viewstruct>
```

Liens entre ViewStruct : viewstruct_link

Sur une url, nous ne faisons plus référence à la structure des éventuelles vues destination.

Sur l'url, les seuls paramètres sont les champs publics de la ViewStruct courante.

Aussi faut-il mettre en place une configuration permettant d'établir la correspondance entre les valeurs des champs publics de chaque ViewStruct afin de les connecter entre elles.
"field_match" permet de définir la traduction d'un nom de champ public d'une ViewStruct à une autre. Par défaut, la traduction est de conserver le même nom.

```
<viewstruct_link_def>
    <viewstruct_link src="Commande1" dst="Commande2">
        <field_match src="nomtie" dst="nomtie"/>
    </viewstruct_link>
</viewstruct_link_def>
```

Définition des vues métier : BusinessView

La vue métier est une ViewStruct et une cible.
C'est aussi un des critères qui entre dans le choix d'une feuille de style.

```
<businessview_def>
    <businessview name="visualiserClient" viewstruct="VisuTiers" target="CLI" "/>
    <businessview name="visualiserFournisseur" viewstruct="VisuTiers" target="FOU" executeQuery="optimize"/>
</businessview_def>
```

Choix des feuilles de styles

Le choix d'une feuille de style est l'aboutissement du traitement d'une requête applicative. Cela permettra de choisir la présentation qui sera générée pour l'utilisateur.

Le choix d'une feuille de style s'effectue en fonction de trois critères : Navigateur, Langue, vue métier.

```
<xsl_param rootPath="e-GX/">
    <xsl support="NAV/MIC" lang="FRA" businessview="visuContenuGSOC" fileRef="langue/fra/xsl/Html/GSOC_Contenu.xsl"/>
    <xsl support="NAV/MIC" lang="FRA" businessview="visuUtuti" fileRef="langue/fra/xsl/Html/UAGE_Detail.xsl"/>
    <xsl support="NAV/MIC" lang="FRA" businessview="visuAdrLAPP" fileRef="langue/fra/xsl/Html/GADR_Liste.xsl"/>
    <xsl support="NAV/MIC" lang="FRA" businessview="visuAdrLECR" fileRef="langue/fra/xsl/Html/GADR_Liste.xsl"/>
    <xsl support="NAV/MIC" lang="FRA" businessview="visuAdrLEVE" fileRef="langue/fra/xsl/Html/GADR_Liste.xsl"/>
</xsl_param>
```

Définition des actions : action

Le décodage des paramètres d'une action de cinématique est effectué par un traitement générique.
Cependant, sur l'url, on ne sait pas reconnaître le type des paramètres. C'est pourquoi, la configuration des actions permet de définir pour chaque action la liste des types des paramètres.

Par ailleurs, l'attribut "**class**" spécifie la classe à instancier pour réaliser le traitement correspondant à l'action. C'est le moyen de découpler les noms d'actions de l'url et les noms et packages des classes.

Certains paramètres d'action sont optionnels. Ils sont ordonnés et toujours placés en fin de signature. Pour définir les paramètres d'une action, on renseigne l'élement "required" et/ou l'élément "optional".

Les type de paramètre sont les suivants :

nom	description
BusinessView	Numéro de BusinessView.
View	Nom de View publiée sur la BusinessView courante.
String	Chaîne de caractères.
Pointer	Pointeur sur un attribut de la requête applicative. Le résultat du dé-référencement est une "String".

Le socle fournit des actions de base qui peuvent être divisées en trois catégories :

- les actions liées navigation,
- les actions liées au CRUD,
- les actions liées à l'identification.

Ces actions ne doivent en aucun cas être modifiées (voir le document de référence sur les actions).

Définition des macro-actions

Pour faciliter l'écriture des actions, le socle donne la possibilité de définir des macro-actions. Ce sont des actions qui représentent la combinaison d'autres actions existantes, qui peuvent elles-mêmes être des macro-actions. Le socle décompose la macro action en action unitaire. C'est comme si la cinématique était déjà composée d'actions unitaires.
Voici un exemple de macro action qui contient deux actions : "retrieve" et "sort".
L'écriture de cette macro se fait de la façon suivante :

```
<action name="init">
    <required>
        <param type="BusinessView" name="businessView"/>
    </required>
    <optional>
        <param type="String" name="where"/>
        <param type="String" name="orderBy"/>
    </optional>
    <action_ref name="retrieve">
        <param_ref name="businessView" ref="businessView"/>
        <param_ref name="key" ref="where"/>
    </action_ref>
    <action_ref name="sort">
        <param_ref name="businessView" ref="businessView"/>
        <param_ref name="key" ref="orderBy"/>
    </action_ref>
</action>
```

Les différences avec une action classique :

- La macro action "init" ne correspond pas à une classe java ;
- La macro action définit un nœud **action_ref** pour chaque action composant la macro action.
- Sous ce nœud **action_ref** sont définis autant de nœud **param_ref** qu'il y a de paramètres à passer à l'action composante.
- **action_ref** : **name** : nom dans la configuration de l'action composante. Peut elle-même être une macro-action. Ces actions composantes seront exécutées en suivant l'ordre de déclaration.
- **param_ref** : chaque paramètre à passer à **l'action_ref**. **name** = nom du paramètre de l'action_ref, **ref** = nom du paramètre de la macro-action correspondant.

Lors de la décomposition de la macro en actions unitaires, la cohérence des types de paramètres est contrôlée entre la macro et les actions composantes.
Il y a une exception à ce contrôle, cela concerne le type Pointer. Ce paramètre est remplacé par sa valeur qui est un attribut applicatif contenu dans la requête.

Couche Métier – paramètres des composants

Définition des ViewObject

La définition des ViewObject permet de créer des références utilisables par la configuration pour désigner des ViewObject Oracle.

La ViewObject est une composante difficile à cerner dans l'architecture e-GX car elle se trouve à la limite de l'Application et du Métier. C'est l'itérateur d'objet métier. A ce titre, elle est utilisée dans l'application, mais est fortement dépendante des objets métier qu'elle référence.

Il se trouve que c'est l'instanciation du Document qui demande l'instanciation d'une ViewObject associée. Alors que l'instanciation de Document se réalise dans la couche "Application", cette instanciation de ViewObject se réalise, en fait, dans la couche métier.

Du point de vue BC4J, la notion de VienLink existe aussi. Rien n'empêche de définir des ViewLink-Oracle sur une ViewObject.

Les différents attributs disponibles sur une ViewObject sont résumés dans le tableau qui suit.

Attribut	Fonction	Obligatoire
name	Nom e-GX de la ViewObject.	X
defFullName	Nom de la ViewObject Oracle associée, chemin de package compris.	X
depthCount	Profondeur de lecture des ViewLink Oracle que le socle technique doit prendre en compte, si des ViewLink-Oracle sont définis sur la ViewObject concernée. La valeur par défaut de "depthCount" est 5.	
maxFetchSize	Le nombre d'enregistrements maximum lu à chaque accès base.	
minInstance	Nombre d'instances pré-instanciées dans le pool de ViewObject.	
maxInstance	Nombre maximun d'instances possibles pour cette ViewObject.	

Trois attributs sont obligatoires sur `viewobject_def` :

minInstance et maxInstance.

Ce sont les valeurs par défaut qui s'appliqueront à toutes les ViewObject concernant la gestion de leur pool. Ils définissent respectivement le nombre minimal d'instances et le nombre maximal d'instances.

Cette valeur par défaut peut être modifiée spécifiquement pour chaque ViewObject en renseignant les même attributs directement dans l'élément de définition de la ViewObject .

L'attribut minInstance définit un nombre d'objets pré-instanciés. Il faut savoir que si on définit une valeur élevée, le démarrage d'e-GX sera ralenti, car il faudra construire beaucoup d'objets. Pas de panique, ce chargement à lieu la première fois qu'on une instance d'e-GX, pas à chaque nouvelle session utilisateur.

maxFetchSize

Fixe le nombre maximum d'enregistrements ramenés à chaque requête. Sorte de fusible pour éviter de consommer trop de ressources et d'écrouler les performances du serveur. "maxFetchSize" ne peux pas être nul.

```
<viewobject_def minInstance="1" maxInstance="5" maxFetchSize="200"/>
    <viewobject name="JTieView" defFullName="fr.generix.metier.JTieView" depthCount="0" maxFetchSize="50"/>
    <viewobject name="JEveView" defFullName="fr.generix.oracle.JEveView" depthCount="0"/>
    <viewobject name="JPosteView" defFullName="fr.generix.oracle.JPosteView" depthCount="0"/>
</viewobject_def>
```

Quand on positionne la valeur de defFullName à "DYNAMIC", cela signifie qu'il s'agit d'une ViewObject dynamique. Dans ce cas, cette ViewObject devra être associée à un `DynRowDocument`.

Définition des ApplicationModule

Nous sommes dans le contexte d'architecture métier suivant :

- Un ApplicationModule SessionMetier est le seul ApplicationModule root de la couche métier.
- Tous les autres ApplicationModules sont "nested" de SessionMetier.
- L'instanciation des "nested" ApplicationModule de SessionMetier est dynamique.

Corollaires :

- Tous les "nested" ApplicationModule sont "locaux".
- On ne peut pas déployer les nested ApplicationModule comme EJB.

Deux attributs sont obligatoires sur `applicationmodule_def`. Il s'agit de `minInstance` et `maxInstance`.

Ce sont les valeurs par défaut qui s'appliqueront à tous les ApplicationModules concernant la gestion de leur pool. Ils définissent respectivement le nombre minimal d'instances et le nombre maximal d'instances.

Cette valeur par défaut peut être modifiée spécifiquement pour chaque ApplicationModule en renseignant les même attributs directement dans l'élément de définition de l'ApplicationModule.

L'attribut minInstance définit un nombre d'objets pré-instanciés. Il faut savoir que si on définit une valeur élevée, le démarrage d'e-GX sera ralenti, car il faudra construire beaucoup d'objets. Pas de panique, ce chargement à lieu la première fois qu'on une instance d'e-GX, pas à chaque nouvelle session utilisateur.

```
<applicationmodule_def minInstance="0" maxInstance="10">
    <applicationmodule sessionType="statefull" defFullName="fr.generix.metier.bc4j.appel.visualiser"/>
    <applicationmodule sessionType="statefull" defFullName="fr.generix.metier.bc4j.service.visualiser"/>
    <applicationmodule sessionType="statefull" defFullName="fr.generix.metier.bc4j.encours.GestionEncours"/>
    <applicationmodule sessionType="statefull" defFullName="fr.generix.metier.bc4j.tiers.visualiser" maxInstance="15"/>
</applicationmodule_def>
```

Schema XML

Le fichier de référence est :
\\alcis\e-GX\developpements\socletechnique\V4.4.0\public\document\configuration_440.xsd

Règle de nommage du XML de configuration

noms d'éléments :

- que des minuscules
- on sépare les mots par des '_' (underscore). remarque : la notion de mot peut paraître floue parfois. Par exemple "businessview" en un seul mot, car c'est un terme de la conception (autre exemple : "viewstruct_link_def").

noms d'attributs :

- première lettre en minuscule.
- majuscules pour la première lettre des mots suivants. (pas de underscore entre les mots).

noms de type complexe ou de groupes d'attributs (spécifique XML Schema) :

- commencent toujours par une majuscule
- majuscules pour la première lettre des mots suivants. (pas de underscore entre les mots).

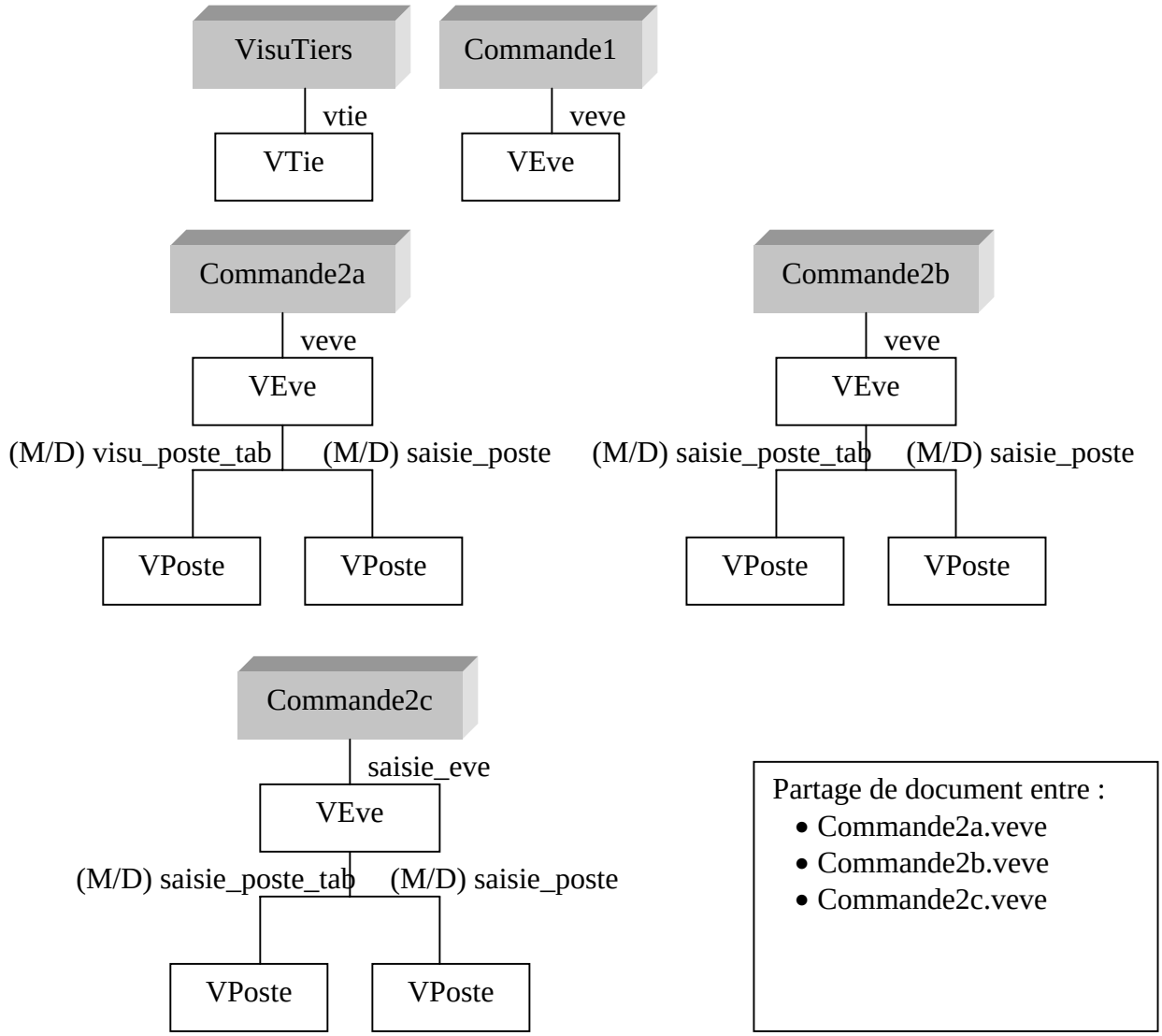
Exemples

Ce paragraphe pratique permet de répondre aux interrogations concrètes en présentant des exemples résolus de configuration ou des pistes par rapport à un cas donné.

Saisie de commande

C'est un cas "classique" de configuration.

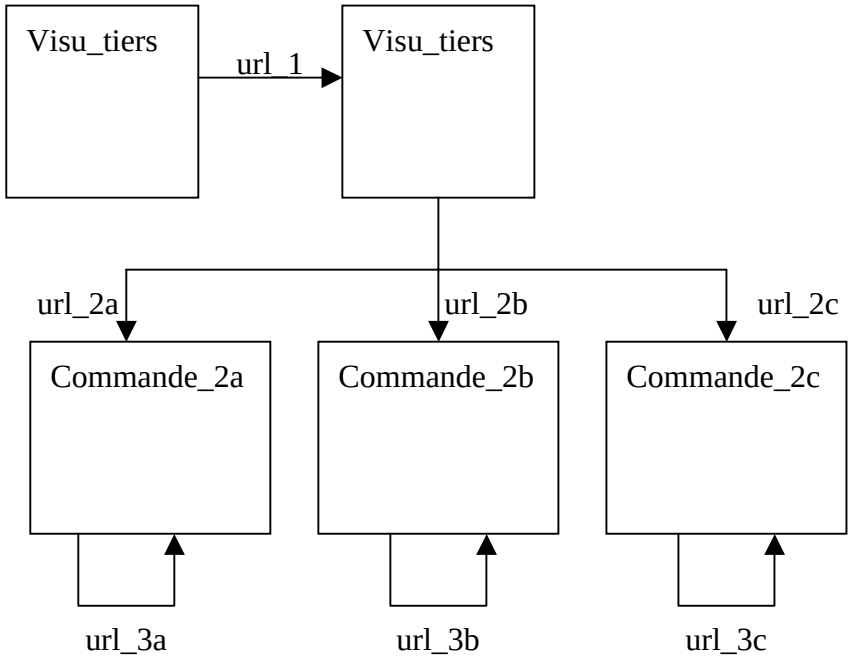
Soit les structures de vues suivantes :



Partage de document entre :

- Commande2a.veve
- Commande2b.veve
- Commande2c.veve

Soient les navigations suivantes :



Ce qui donne le fichier de configuration suivant :

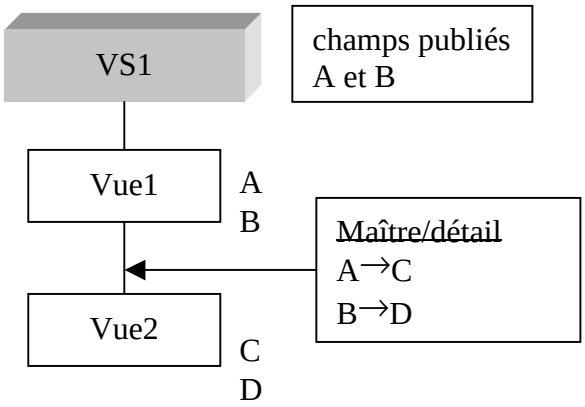
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by  () -->
<!--Tiens compte de la notion de champs publiés sur une ViewStruct et de clause retrieve order by par défaut associées à un type de Vue-->
<?xml version="1.0" encoding="iso-8859-1"?>
<config xsi:noNamespaceSchemaLocation="..\configuration.xsd" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
    <application>
        <document_def>
            <document name="DocTie" class="fr.generix.tecnicalframework.application.document.RoxDocumentImpl" viewobject="JTieView"/>
            <document name="DocEve" class="fr.generix. .tecnicalframework.application.document.RoxDocumentImpl " viewobject="JEveView"/>
            <document name="DocPoste" class="fr.generix.metier.document.DocSpecifique1" viewobject="JPosteView"/>
        </document_def>
        <view_def>
            <view_type name="VTie" document="DocTie"/>
            <view_type name="VEve" document="DocEve"/>
            <view_type name="VPoste" document="DocPoste">
                <retrieve>
                    <where_clause key="0" value="POSTE='@'" resolverClass="fr.generix.technicalframework.application.DefaultWhereClauseResolver"/>
                    <where_clause key="1" value="POSTE like '@%' AND TOTO &lt; 2'" resolverClass="fr.generix.technicalframework.application.DefaultWhereClauseResolver"/>
                </retrieve>
                <sort>
                    <orderby_clause key="0" value="POSTE ASC, AUTRE DESC"/>
                </sort>
            </view_type>
        </view_def>
        <viewstruct_def>
            <viewstruct name="VisuTiers">
                <view name="vtie" type="VTie" nbline="10" create="false">
```

```
            <field>
                <attribute name="nomtie" public="nomtie"/>
            </field>
        </view>
    </viewstruct>
    <viewstruct name="Commande1">
        <view name="veve" type="VEve" nbline="1" create="false">
            <field>
                <attribute name="nomtie" public="nomtie"/>
                <attribute name="date" public="date"/>
            </field>
        </view>
    </viewstruct>
    <viewstruct name="Commande2a">
        <view name="veve" type="VEve" nbline="1" create="false" docSharedName="entete_commande">
            <field>
                <attribute name="nomtie" public="nomtie"/>
                <attribute name="date" public="date"/>
            </field>
            <view name="visu_poste_tab" type="VPoste" nbline="5" create="false">
                <view_link>
                    <retrieve publicKey="0" viewKey="0">
                        <param domain="business" field="poste"/>
                    </retrieve>
                    <sort publicKey="0" viewKey="0"/>
                </view_link>
            </view>
            <view name="saisie_poste" type="VPoste" nbline="1" create="false">
                <view_link>
                    <retrieve publicKey="0" viewKey="0">
                        <param domain="business" field="poste"/>
                    </retrieve>
                    <sort publicKey="0" viewKey="0"/>
                </view_link>
                <field>
                    <attribute name="codpro" public="codpro"/>
                    <attribute name="qte" public="qte"/>
                </field>
            </view>
        </view>
    </viewstruct>
    <viewstruct name="Commande2b">
        <view name="veve" type="VEve" nbline="1" create="false" docSharedName="entete_commande">
            <view name="saisie_poste_tab" type="VPoste" nbline="3" create="false">
                <field>
                    <attribute name="codpro" public="codpro_tab"/>
                    <attribute name="qte" public="qte_tab"/>
                </field>
            </view>
            <view name="saisie_poste" type="VPoste" nbline="1" create="false">
                <field>
                    <attribute name="codpro" public="codpro"/>
                    <attribute name="qte" public="qte"/>
                </field>
            </view>
        </view>
    </viewstruct>
    <viewstruct name="Commande2c">
        <view name="saisie_eve" type="VEve" nbline="1" create="false" docSharedName="entete_commande">
            <field>
                <attribute name="nomtie" public="nomtie"/>
                <attribute name="date" public="date"/>
            </field>
            <view name="saisie_poste_tab" type="VPoste" nbline="5" create="false">
                <field>
                    <attribute name="codpro" public="codpro_tab"/>
                    <attribute name="qte" public="qte_tab"/>
                </field>
            </view>
            <view name="saisie_poste" type="VPoste" nbline="1" create="false">
```

```
                        <field>
                            <attribute name="codpro" public="codpro"/>
                            <attribute name="qte" public="qte"/>
                        </field>
                    </view>
                </view>
            </viewstruct>
        </viewstruct_def>
        <viewstruct_link_def>
            <viewstruct_link src="VisuTiers" dst="Commande1">
                <field_match src="nomtie" dst="nomtie"/>
            </viewstruct_link>
            <viewstruct_link src="Commande1" dst="Commande2a">
                <field_match src="nomtie" dst="nomtie"/>
                <field_match src="date" dst="date"/>
            </viewstruct_link>
            <viewstruct_link src="Commande1" dst="Commande2b">
                <field_match src="nomtie" dst="nomtie"/>
                <field_match src="date" dst="date"/>
            </viewstruct_link>
            <viewstruct_link src="Commande1" dst="Commande2c">
                <field_match src="nomtie" dst="nomtie"/>
                <field_match src="date" dst="date"/>
            </viewstruct_link>
        </viewstruct_link_def>
        <businessview_def>
            <businessview name="visualiserClient" viewstruct="VisuTiers" target="CLI"/>
            <businessview name="visualiserFournisseur" viewstruct="VisuTiers" target="FOU"/>
            <businessview name="saisieCommande1" viewstruct="Commande1" target="CLI"/>
            <businessview name="saisieCommande2a" viewstruct="Commande2a" target="CLI"/>
            <businessview name="saisieCommande2b" viewstruct="Commande2b" target="CLI"/>
            <businessview name="saisieCommande2c" viewstruct="Commande2c" target="CLI"/>
        </businessview_def>
        <xsl_param rootPath="F:/e-GX/">
            <xsl support="NAV/MIC" lang="FRA" businessview="visualiserClient" fileRef="langue/fra/xsl/Html/visuClient.xsl"/>
            <xsl support="NAV/MIC" lang="FRA" businessview="visualiserFournisseur" fileRef="langue/fra/xsl/Html/visuFournisseur.xsl"/>
            <xsl support="NAV/MIC" lang="FRA" businessview="saisieCommande1" fileRef="langue/fra/xsl/Html/saisComm1.xsl"/>
            <xsl support="NAV/MIC" lang="FRA" businessview="saisieCommande2a" fileRef="langue/fra/xsl/Html/saisComm2a.xsl"/>
            <xsl support="NAV/MIC" lang="FRA" businessview="saisieCommande2b" fileRef="langue/fra/xsl/Html/saisComm2b.xsl"/>
            <xsl support="NAV/MIC" lang="FRA" businessview="saisieCommande2c" fileRef="langue/fra/xsl/Html/saisComm2c.xsl"/>
        </xsl_param>
        <link_to_business>
            <jndi deploy_platform="local" authentication="non_ssl_login" port="2481" host="" sid=""/>
            <business_session sessionType="statefull" defFullName="" connection="" lockingMode="pessimistic"/>
        </link_to_business>
    </application>
    <business>
        <viewobject_def maxFetchSize="200" minInstance="1" maxInstance="20">
            <!-- Ce tag viewobject_def peut etre evite, s il y a beaucoup plus de ViewObject que de Documents-->
            <viewobject name="JTieView" defFullName="fr.generix.metier.JTieView"/>
            <viewobject name="JEveView" defFullName="fr.generix.oracle.JEveView"/>
            <viewobject name="JPosteView" defFullName="fr.generix.oracle.JPosteView"/>
        </viewobject_def>
    </business>
</config>
```

Exemple de Maître/détail

Soit une ViewStruct VS1 publiant deux champs (A et B).
Ces champs publiés correspondent aux champs A et B de la View Vue1 (première View).
On veut définir un maître/détail entre la View "Vue1" et la View "Vue2".



Les parties en gras, de l'exemple suivant, sont importantes pour la réalisation d'un maître/détail.
une clause retrieve réalisant une sélection sur des champs de la vue de détail.
lors de la définition du Vie

Exemple de configuration pour ce cas :

```
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by  () -->
<config
xsi:noNamespaceSchemaLocation="X:\developpements\socletechnique\V2\documents\conception\configuration\configuration.xsd"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
    <application>
        <properties connection="scott/tiger@machine:1521:gnx" cacheActive="true" traceLevel="0" frameMaxNum="10"
defaultExceptionManager="fr.generix.application.ExceptionManager">
            <visible bc4jViewLink="true" field="true"/>
        </properties>
        <viewobject_def>
            <viewobject name="vo1" defFullName="fr.generix.appli.ViewObject1"/>
            <viewobject name="vo2" defFullName="fr.generix.appli.ViewObject2"/>
        </viewobject_def>
        <document_def>
            <document name="doc1" class="fr.generix.application.DocGeneric" viewobject="vo1"/>
            <document name="doc2" class="fr.generix.application.DocGeneric" viewobject="vo2"/>
        </document_def>
        <view_def>
            <view_type name="Vue1" document="doc1">
                <retrieve>
                    <where_clause key="0" value="champA='@' AND champB='@'"
resolverClass="fr.generix.technicalframework.application.DefaultWhereClauseResolver"/>
                </retrieve>
            </view_type>
            <view_type name="Vue2" document="doc2">
                <retrieve>
                    <where_clause key="0" value="champC = '@%' AND champD = '@%'"
resolverClass="fr.generix.technicalframework.application.DefaultWhereClauseResolver"/>
                </retrieve>
            </view_type>
        </view_def>
        <viewstruct_def>
            <viewstruct name="VS1" defaultRetrieve="zero">
                <view type="Vue1" nbline="1">
                    <view_link>
                        <retrieve publicKey="zero" viewKey="0">
                            <param domain="public" field="A"/>
                            <param domain="public" field="B"/>
                        </retrieve>
                    </view_link>
                    <field>
                        <attribute name="A" public="A"/>
                        <attribute name="B" public="B"/>
```

```
                    </field>
                    <view type="Vue2" nbline="5">
                        <view_link>
                            <retrieve publicKey="zero" viewKey="0">
                                <param domain="business" field="A"/>
                                <param domain="business" field="B"/>
                            </retrieve>
                        </view_link>
                    </view>
                </view>
            </viewstruct>
        </viewstruct_def>
        <businessview_def>
            <businessview name="" viewstruct="" target=""/>
        </businessview_def>
        <xsl_param rootPath="">
            <xsl support="NAV/MIC" lang="FRA" businessview="" fileRef=""/>
        </xsl_param>
        <action_def>
        …
        </action_def>
        <link_to_business>
            <jndi deployPlatform="local" authentication="non_ssl_login"/>
            <business_session sessionType="statefull" defFullName="" connection="" lockingMode="pessimistic"/>
        </link_to_business>
        <link_to_presentation>
        …
        </link_to_presentation>
    </application>
    <business>
        <viewobject_def maxFetchSize="200" minInstance="1" maxInstance="20">
            <viewobject name="vo1" defFullName="fr.generix.appli.ViewObject1"/>
            <viewobject name="vo2" defFullName="fr.generix.appli.ViewObject2"/>
        </viewobject_def>
    </business>
</config>
```