NVDCMS 2.0

Guide du développeur

NVDCMS 2.0: Guide du développeur

Table des matières

Chapitre 1. Introduction

Ce manuel s'adresse aux développeurs de sites à contenu dynamique. Il présente les différentes fonctionnalité du language de la plate-forme. Une explication détaillée de chaque instruction du langage est disponible dans la "Documentation de référence du langage".

Pour l'installation logicielle, il est recommandé de se reporter au "Guide d'administration"

Pour simplifier l'écriture des exemples de ce manuel, nous prenons par défaut le préfixe de namespace XML "nvd:" pour écrire les instructions du langage.

1.1. Prérequis

Une connaissance du langage XML est indispensable pour prendre en main l'utilisation du language NVD-CMS. De plus il est préférable d'avoir de bonnes notions d'XSL (non expliqué dans ce manuel) pour obtenir une présentation dans différents formats (HTML, FO, SVG) des documents. Une connaissance de la norme SOAP est par ailleurs nécessaire pour pouvoir implémenter des procédures distantes.

Les différentes spécifications des normes sont disponibles online sur le site du W3C :

- XML : http://www.w3c.org/XML/
- XSL : http://www.w3c.org/Style/XSL/
- SVG : http://www.w3.org/Graphics/SVG/
- FO : http://www.w3.org/TR/xsl/slice6.html#fo-section

Chapitre 2. Hello World

Nous allons présenter dans ce chapitre un exemple sur la réalisation d'un site web simple.

La création d'un site web simple se décompose en plusieurs étapes :

- La définition, à l'aide du "WebManager", des structures de données pour le contenu.
- La gestion du contenu à l'aide du langage NVD-CMS.
- La présentation du contenu à l'aide de la feuille de style XSL.

Dans l'exemple ci dessous, nous allons décrire les pages XML qui définissent et indiquent le contenu utilisé, ainsi que la feuille de style XSL sui sera appliquée au document afin de générer la page HTML correspondante.

Astuce

L'interface "WebManager" simplifie le processus de création de structure de données et dispose d'outils de base pour la saisie du contenu. Cependant il est toujours possible de redéfinir ses propres interfaces en cas de nécessité.

Dans cet exemple nous allons développer une page principale qui affiche un texte simple. Le texte sera saisi à l'aide d'un formulaire. Nous ne détaillerons pas la partie qui concerne le langage XSL (se référer à une documentation adéquate).

Dans les exemples suivants, nous supposerons que l'utilisateur dispose d'un site Web sur la plate-forme et qu'il est authentifié comme administrateur du site.

2.1. Structure de données

Les structures de données sont représentés sous forme d'objets composés de champs typés. La structure qui contiendra notre message est un objet simple que nous appellerons 'message', ne comprend qu'un seul champ que nous appellerons "texte". Ce champs est d'une longueur de 1024 caractères.

La création d'un objet peut s'effectuer à l'aide de l'interface "WebManager". Nous allons cependant utiliser un document XML composé de deux commandes NVD-CMS, une pour créer l'objet 'message', et l'autre pour créer le champ 'texte'.

```
<?xml version='1.0' encoding='UTF-8'?>
<?nvd-process name="document-process" prefix="nvd" ?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">

<nvd:command name="object.create" >
<nvd:parameter name="name">message</nvd:parameter>
<nvd:result name="id" variable="object-id" />
</nvd:command>

<nvd:command name="object-field.create" object-name="message">
<nvd:parameter name="name">texte</nvd:parameter>
<nvd:parameter name="datatype">string</nvd:parameter>
<nvd:parameter name="minimum" type="integer">0</nvd:parameter>
<nvd:parameter name="maximum" type="integer">1024</nvd:parameter>

<nvd:parameter name="policy-create">anyone</nvd:parameter>
<nvd:parameter name="policy-modify">anyone</nvd:parameter>
<nvd:parameter name="mandatory">false</nvd:parameter>
<nvd:parameter name="indexed">false</nvd:parameter>
</nvd:command>
```

```
</page>
```

A noter ici, nous déclarons le namespace ayant comme préfixe "nvd" pour l'exécution des instructions : *xmlns:nvd="http://www.nvdcms.org/cms/"*. De plus nous indiquons dans le processing instruction *<?nvd-process name="document-process" prefix="nvd" ?>* que tous les élements avec le préfixe "nvd" devront être interprétés comme des instructions à exécuter.

Nous créons ensuite une instance de l'objet 'message' contenant le texte initiale suivant : "Hello World".

```
<?xml version='1.0' encoding='UTF-8'?>
<?nvd-process name="document-process" prefix="nvd" ?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">

<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte">Hello World</nvd:parameter>
</nvd:command>

</page>
```

Les deux documents ci-dessus ne doivent être exécutés qu'une seule fois, afin de ne créer qu'une seule instance de l'objet.

2.2. Définition du contenu de la page principale

Nous allons maintenant récuperer le contenu de l'instance que nous venons de créer à l'aide d'un document comprenant une instruction de lecture de données.

```
<?xml version='1.0' encoding='UTF-8'?>
<?nvd-process name="document-process" prefix="nvd" ?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">

<message>
<nvd:instance object-name="message">
<nvd:value-of select="@texte"/>
</nvd:instance>
</message>
</page>
```

Après exécution, le document obtenu en sortie est le suivant :

```
<?xml version='1.0' encoding='UTF-8'?>
<?nvd-process name="document-process" prefix="nvd" ?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">

<message>
Hello World
</message>
</page>
```

L'instruction "nvd:instance" demande à l'aide de l'attribut "object-name", la liste des instances de l'objet 'message'. L'instruction "nvd:value" indique que nous souhaitons récuperer le contenu du champ 'texte' de l'instance, en l'occurence "Hello World". Le caractère '@' précède le nom du champ "texte" dans l'instruction *<nvd:value-of select="@texte"/>* pour préciser qu'il s'agit d'un champ de l'objet (et non pas d'un nom réservé par le langage NVD-CMS).

Il ne nous reste plus qu'à appliquer une feuille de style XSL au document pour obtenir une présentation au format HTML.

2.3. Réalisation de la feuille de style XSL

La feuille de style XSL dénifit les régles de transformation du document XML. Dans le cas qui nous intéresse nous allons définir la présentation HTML de notre document XML à savoir :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output media-type="text/html" method="html"  encoding="ISO-8859-1" />

<xsl:template match="page">
<HTML>
<HEAD>
</HEAD>
<BODY>
<xsl:value-of select="message"/>
</BODY>
</HTML>
</xsl:template>

</xsl:stylesheet>
```

La déclaration *<xsl:output media-type="text/html" method="html" encoding="ISO-8859-1" />* indique que nous souhaitons une sortie au format "text/html". La définition de la présentation de l'élément "page" de notre document XML est fait à l'aide des règles définis dans l'élément *<xsl:template match="page">...</xsl:template>*

La feuille de style XSL est enregistrée sous le nom "index.xsl". Pour indiquer dans le document XML le fichier XSL à utiliser pour la transformation nous ajoutons le "processing instruction" suivant :

```
<?nvd-process name="xsl-transform" href="index.xsl"?>
```

Le serveur est alors en mesure d'effectuer la transformation du document à l'aide du fichier XML/XSL "index.xsl" et de générer la sortie HTML correspondante.

Astuce

Certains navigateurs Web sont capables d'effectuer la transformation XSL. Dans notre exemple, pour indiquer la transformation du coté navigateur il suffit de remplacer le processing instruction "nvd-process" par :

```
<?xml-stylesheet href="index.xsl" type="text/xsl"?>
```

2.4. Dynamisation du document

Afin de dynamiser le contenu de la page, nous allons ajouter un petit formulaire de saisi du contenu du message. Nous allons donc modifier la structure du document XML et ajouter une règle de présentation XSL pour le formulaire en HTML.

Nous commençons par ajouter le formulaire de saisie dans la feuille de style XSL.

```
<xsl:template match="formulaire">
<form method="post" action="?">
<input name="modify" type="hidden" value="true"/>

<input name="instance-id" type="hidden">
<xsl:attribute name="value">
<xsl:value-of select="instance-id"/>
</xsl:attribute>
</input>

Votre message : <input type="text" name="message" size="36"/>

<input type="submit"/>
```

```
</form>
</xsl:template>
```

Nous ajoutons maintenant dans le document XML l'instruction prennant en compte la requête du formulaire.

```
<nvd:if test="$parameter.modify == 'true'">
<nvd:command name="instance.modify" id="{$parameter.instance-id}">
<nvd:parameter name="texte"><nvd:variable exec-priority="1"
select="parameter.message"/></nvd:parameter>
</nvd:command>
</nvd:if>
```

Le contenu de la variable "parameter.message" correspond à la valeur saisie dans le formulaire à l'aide de la déclaration *<input type="text" name="message" size="36"/>* après validation.

Nous utilisons un traitement conditionnel qui teste s'il y a bien eu un envoi du contenu du formulaire HTML. La variable "parameter.modify" contient 'true' si le formulaire a bien été reçu. Sa valeur est testée dans l'instruction *<nvd:if test="$parameter.modify == 'true'">*. Si la condition n'est pas valable, la commande dans l'élément "nvd:if" ne sera pas exécutée.

Si la commande est exécuté, le champ 'message' de l'instance sera modifié.

Nous devons maintenant compléter le document XML en y ajoutant l'élément qui sera transformé en formulaire après application de la feuille de style XSL. Par ailleurs il est nécessaire d'y ajouter l'identifiant (id) de l'instance de l'objet 'message' (qui contient à l'origine le texte "Hello World") à modifier.

```
<formulaire>
<instance-id><nvd:instance object-name="message"><nvd:value-of
select="id"/></nvd:instance></instance-id>
</formulaire>
```

Au final cela nous utilisons le document XML suivant :

```
<?xml version='1.0' encoding='UTF-8'?>
<?nvd-process name="document-process" prefix="nvd" ?>
<?nvd-process name="xsl-transform" href="index.xsl"?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">

<nvd:if test="$parameter.modify == 'true'">
<nvd:command name="instance.modify" id="{$parameter.instance-id}">
<nvd:parameter name="texte"><nvd:variable exec-priority="1"
select="parameter.message"/></nvd:parameter>
</nvd:command>
</nvd:if>


<message>
<nvd:instance object-name="message">
<nvd:value-of select="@texte"/>
</nvd:instance>
</message>


<formulaire>
<instance-id><nvd:instance object-name="message"><nvd:value-of
select="id"/></nvd:instance></instance-id>
</formulaire>

</page>
```

Avec la feuille de style XSL suivante :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output media-type="text/html" method="html"  encoding="ISO-8859-1" />

<xsl:template match="page">
<HTML>
<HEAD>
</HEAD>
<BODY>
<xsl:value-of select="message"/>
<xsl:apply-templates select="formulaire"/>
</BODY>
</HTML>
</xsl:template>


<xsl:template match="formulaire">
<form method="post" action="?">
<input name="modify" type="hidden" value="true"/>

<input name="instance-id" type="hidden">
<xsl:attribute name="value">
<xsl:value-of select="instance-id"/>
</xsl:attribute>
</input>

Votre message : <input type="text" name="message" size="36"/>

<input type="submit"/>
</form>
</xsl:template>

</xsl:stylesheet>
```

Chapitre 3. Langage NVD-CMS

3.1. Introdution

Le langage NVD-CMS est un langage XML, pour la gestion de contenu. L'appel des instructions s'effectue à l'aide d'éléments définis dans le namespace XML "http://www.nvdcms.org/cms/". L'exécution des instructions s'effectue en ordre descendant dans le document. Cependant plusieurs passes d'exécution peuvent être effectuées pour des traitements plus complexes dont les resultats sont interdépendants.

Le language dispose de "processing instruction" qui définissent entre autres les différentes types d'exécutions du document.

L'essentiel des opérations consiste à manipuler des structures de données en lecture et en écriture. La plate forme permet de définir des objets qui peuvent être instanciés (et/ou modifiés) et placés dans des listes de publication.

3.1.1. Structure de données

Le logiciel NVD-CMS permet de manipuler le contenu et les structures de données propre à chaque site. Un administrateur de site peut gérer les différentes structures. Les structures de données sont définies à l'aide d'objet regroupant des champs dont on spécifie le nom et le type.

Le langage permet de créer des structures à l'aide de commande. La création d'un structure se décompose en 2 étapes : la création de l'objet et la création des champs de l'objet.

Les structures de données sont créées à l'aide d'instructions du langage. La création de l'objet avec un nom donné nécessite l'exécution de la commande "object.create".

```
<nvd:command name="object.create" >
<nvd:parameter name="name">message</nvd:parameter>
<nvd:result name="id" variable="object-id" />
</nvd:command>
```

Une fois l'objet créé il est nécessaire de compléter sa description en indiquant les différents champs qui le composent. Chaque champ est typé (voir Annexe A. *Type de données des objets*). Leur contenu peut être indexé dans le moteur de recherche interne. De plus il est possible d'indiquer s'il est obligatoire de renseigner un champ pour valider la création complète de l'instance (notion de champs obligatoires), ainsi que les privilèges requis pour pouvoir créer ou modifier le champ de l'instance.

La création d'un champ utilise la commande "object-field.create" du langage.

```
<nvd:command name="object-field.create" object-name="message">
<nvd:parameter name="name">texte</nvd:parameter>
<nvd:parameter name="datatype">string</nvd:parameter>
<nvd:parameter name="minimum" type="integer">0</nvd:parameter>
<nvd:parameter name="maximum" type="integer">1024</nvd:parameter>

<nvd:parameter name="policy-create">anyone</nvd:parameter>
<nvd:parameter name="policy-modify">anyone</nvd:parameter>
<nvd:parameter name="mandatory">false</nvd:parameter>
<nvd:parameter name="indexed">false</nvd:parameter>
</nvd:command>
```

Dans cet exemple, le champ 'texte' de l'objet 'message' est créé avec un type de données chaine de caractères avec une taille maximale de 1024 caractères. Il n'est pas nécessaire de renseigner le champ pour créer une instance complète (mandatory : false). Le contenu du champ n'est pas indéxé (indexed : false). Le privilège requis pour la création du contenu dans ce champ est 'anyone', idem pour la modification.

3.1.2. Instructions

Le langage NVD-CMS utilise des instructions (ou modules) en XML. En fonction de l'opération requise le nom de l'élément à utiliser est différent. Par exemple, pour lister des instances, l'instruction "nvd:instance" est utilisée, pour lister des publication, "nvd:publication", et pour afficher les nom de domaine associé à un site, "nvd:domain".

Par exemple, nous listons les instances de l'objet message à l'aide de l'instruction suivante :

```
<nvd:instance object-name="message" limit="100">
<nvd:value-of select="@texte"/>
</nvd:instance>
```

L'utilisation du préfixe "nvd:" n'est pas imposée. Le préfixe doit être associé au namespace "http://www.nvdcms.org/cms/" pour être valide. De plus, pour que le code soit exécuté, il est nécessaire d'indiquer chaque préfixe à exécuter dans un "processing instruction".

De manière générale, toutes les instructions fonctionnent avec des attributs commun comme "limit" ou "start" qui permettent de définir la taille de la liste de valeurs à recupérer.

La récupération d'une valeur dans une liste est effectuée à l'aide de l'élément "nvd:value-of". Celui-ci indique à l'aide de l'attribut "select", le nom de la valeur à récupérer. Dans l'exemple ci-dessus *nvd:value-of select="@texte"* indique que nous souhaitons récupérer la valeur "@texte" (où '@' precise qu'il s'agit d'un champ de l'objet).

Lorsqu'une liste de valeur est recupérée, pour chaque item, le contenu de l'élement instruction est cloné. Avec l'exemple suivant :

```
<nvd:instance object-name="message" limit="3">
<item>
<nvd:value-of select="@texte"/>
</item>
</nvd:instance>
```

Nous obtenons après traitement, le résultat suivant (si les instances de l'objet 'message' existe) :

```
<item>
Texte1
</item>
<item>
Texte2
</item>
<item>
Texte3
</item>
```

"Texte1", "Texte2" et "Texte3" sont les contenus des champs 'message' des 3 instances récupérées.

3.1.3. Commandes

Les commandes sont des instructions qui bénéfient de fonctionnalités spécifiques pour les traitrements en insertion de données. La saisie de paramètres typés et "multiples", la remontéz d'erreurs, ou l'enchainement de traitements dans le cadre d'une transaction sont prises en compte par les commandes.

La syntaxe d'une commande diffère de celle des instructions simple dans la mesure où elle est capable de traiter avec précision les paramètres passés à l'aide des éléments "nvd:parameter". Dans l'exemple de création d'instance suivant, pour l'objet 'message', nous passons au paramètre "texte" la valeur "Hello World".

```
<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte">Hello World</nvd:parameter>
</nvd:command>
```

3.1.4. processing instruction

Les processing instructions indique les différentes opérations à effectuer sur un document. Les processing instructions les plus fréquemment employés dans un document sont :

```
<?nvd-process name="document-process" prefix="nvd" ?>
<?nvd-process name="xsl-transform" href="index.xsl"?>
```

Le processing instruction "document-process" indique une execution des instructions dans le namespace "http://www.nvdcms.org/cms/" ayant pour préfixe "nvd". Le processing instruction "xsl-transform" applique la feuille de style XSL contenu dans le fichier "index.xsl".

Les "processing instructions" sont exécutés dans leur ordre d'apparition dans le document.

3.2. Exécution du document

3.2.1. Exécution simple

Les instructions du langage NVD-CMS doivent être définies au travers des éléments d'un namespace ayant comme URL "http://www.nvdcms.org/cms/". Le préfixe de namespace (en général "nvd") doit être indiqué avec un 'processing instruction' "nvd-process" pour exécuter les instructions.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<?nvd-process name="document-process" prefix="nvd"?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">
<nvd:...></nvd:...>
<nvd:...></nvd:...>
</page>
```

Dans l'exemple ci-dessus, toute les instructions définies dans les éléments avec le préfixe "nvd" sont exécutés dans l'ordre d'appararition dans le document.

3.2.2. Exécution avec priorités

Il est possible d'exécuter des éléments d'un préfixe données en plusieurs étapes, pour réaliser certaines opérations un peu plus complexes.

Un numéro de priorité d'exécution est donné à une instruction avec l'attribut "exec-priority". L'ordre d'exécution est alors du numéro le plus faible au plus élevé, sachant que par défaut, si aucun numéro n'est donné, l'exécution de l'instruction est effectuée en dernier.

Pour un numéro de priorité d'exécution donné, le traitement des instructions est effectué dans leur ordre d'apparation dans le document.

Dans l'exemple ci dessous nous imbriquons deux instructions "nvd:instance".

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<?nvd-process name="document-process" prefix="nvd"?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">
<nvd:instance object-name="message" limit="2" exec-priority="1">
<nvd:value-of select="@texte"/>
<nvd:instance object-name="voiture" limit="1" exec-priority="2">
<nvd:value-of select="@modele"/>
</nvd:instance>
</nvd:instance>
</page>
```

Si on décompose l'exécution, l'exécution de la priorité 1 est effectuée dans un premier temps, ce qui nous donne apres traitement.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<?nvd-process name="document-process" prefix="nvd"?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">

Texte1
<nvd:instance object-name="voiture" limit="1" exec-priority="2">
<nvd:value-of select="@modele"/>
</nvd:instance>

Texte2
<nvd:instance object-name="voiture" limit="1" exec-priority="2">
<nvd:value-of select="@modele"/>
</nvd:instance>

</page>
```

Et après exécution de la priorité 2, nous obtenons :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<?nvd-process name="document-process" prefix="nvd"?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">

Texte1
Modele1

Texte2
Modele1

</page>
```

Généralement les priorités d'exécution sont souvent utilisées pour imbriquer des tests conditionnels, des variables et des commandes.

```
<nvd:if exec-priority="1" test="$parameter.text != null">
<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte"><nvd:variable exec-priority="1" select="parameter.text"/></nvd:parame
</nvd:command>
</nvd:if>
```

Ici, le test conditionnel ("nvd:if") et la sélection de la variable ("nvd:variable") sont effectués dans un premier temps, puis si le test est valide, la commande "instance.create" est exécutée en dernier.

3.2.3. Exécution en plusieurs passes

Il est possible de déclarer plusieurs préfixes d'exécution avec le même namespace à l'aide de 'processing instruction'. Cela permet d'effectuer plusieurs passes d'exécution sur le document.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<?nvd-process name="document-process" prefix="nvd"?>
<?nvd-process name="document-process" prefix="nvd-post"?>

<page
xmlns:nvd="http://www.nvdcms.org/cms/"
xmlns:nvd-post="http://www.nvdcms.org/cms/"
>
<nvd:...></nvd:...>
<nvd-post:...></nvd-post:...>
</page>
```

Les instructions du préfixe "nvd" seront toutes exécutées dans un premier temps puis les instructions avec comme préfixe "nvd-post" seront traitées dans un second temps.

Le principe de fonctionnement est similaire aux ordres d'exécution avec cependant une meilleure lisibilité et la possibilité de définir sans ambiguïté des instructions avec des sous-éléments (comme "value-of" ou "subset") complètement imbriqués.

Dans l'exemple suivant nous imbriquons sans ambiguïté le "value-of" qui traitait des données de l'objet "message" à l'intérieur d'une instruction avec un "value-of" concernant l'objet "voiture".

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<?nvd-process name="document-process" prefix="nvd"?>
<?nvd-process name="document-process" prefix="nvd-post"?>

<page
xmlns:nvd="http://www.nvdcms.org/cms/"
xmlns:nvd-post="http://www.nvdcms.org/cms/"
>
<nvd:instance object-name="message" limit="2" exec-priority="1">
<nvd-post:instance object-name="voiture" limit="1" exec-priority="2">
<nvd:value-of select="@texte"/>
<nvd-post:value-of select="@modele"/>
</nvd-post:instance>
</nvd:instance>
</page>
```

Après traitement du préfixe "nvd" nous obtenons le document suivant :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<?nvd-process name="document-process" prefix="nvd-post"?>

<page
xmlns:nvd="http://www.nvdcms.org/cms/"
xmlns:nvd-post="http://www.nvdcms.org/cms/"
>
<nvd-post:instance object-name="voiture" limit="1" exec-priority="2">
Texte1
<nvd-post:value-of select="@modele"/>
</nvd-post:instance>
<nvd-post:instance object-name="voiture" limit="1" exec-priority="2">
Texte2
<nvd-post:value-of select="@modele"/>
</nvd-post:instance>
</page>
```

Puis le traitement du préfixe "nvd-post" le document final est produit :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<page
xmlns:nvd="http://www.nvdcms.org/cms/"
xmlns:nvd-post="http://www.nvdcms.org/cms/"
>
Texte1
Modele1
Texte2
Modele1
</page>
```

3.2.4. Pipeline d'exécution

Il est possible de définir différent processus d'exécution du document à l'aide de "processing instruction". Comme nous l'avions indiqué plus haut (voir Section 3.1.4), les cas le plus fréquent d'enchainement de processus consiste à effectuer un traitement des instruction du langage NVD-CMS puis une transformation XSL du document. Par exemple :

```
<?nvd-process name="document-process" prefix="nvd" ?>
<?nvd-process name="xsl-transform" href="index.xsl"?>
```

Cependant, une série plus complexe de transformation est obtenue en définissant un "pipeline" d'exécutions de processus.

```
<?nvd-process name="document-process" prefix="nvd" ?>
<?nvd-process name="xsl-transform" href="transform1.xsl"?>
<?nvd-process name="document-process" prefix="nvd" ?>
<?nvd-process name="xsl-transform" href="transform2.xsl"?>
```

Dans cet exemple, une première interprétation dans le langage NVD-CMS est effectuée sur le document, suivie d'une transformation à l'aide fichier "transform1.xsl", elle même suivit d'un nouveau traitement du langage NVD-CMS, avec pour finir, une dernière transformation XSL via le fichier "transform2.xsl".

Dans cet autre exemple, nous utilisons, après traitement des instructions "nvd" une transformation XSL pour produire un document XML FO (Formating Object), suivit de la génération d'un fichier PDF.

```
<?nvd-process name="document-process" prefix="nvd" ?>
<?nvd-process name="xsl-transform" href="index.xsl"?>
<?nvd-process name="fo-transform" format="pdf"?>
```

Si l'attribut "cache-timeout" est utilisé avec un "processing instruction" 'cachable', alors le contenu en sorti après traitement est placé dans un cache pour la durée en seconde spécifiée dans l'attribut.

```
<?nvd-process name="document-process" prefix="nvd" ?>
<?nvd-process name="xsl-transform" href="index.xsl" cache-timeout="60" ?>
```

Ici, le résultat après transformation est conservé en cache pour une durée de 60 secondes.

Il est possible d'ajouter dynamiquement (lors du traitement du document) le "processing instruction" à l'aide la commande "processing-instruction.add".

```
<nvd:command name="processing-instruction.add" instruction=
"nvd-process name=&quot;xsl-transform&quot; href=&quot;index.xsl&quot;"/>
```

Par exemple, pour effectuer la transformation XSLT sur le navigateur si le 'User-Agent' de celui-ci contient "MSIE 6" (cet exemple utilise certaines instructions qui seront détaillées par la suite) :

```
<nvd:nsp>
String s = getVariableAsString( "request.header[User-Agent]" );
if( s.indexOf( "MSIE 6" ) > 0 ) setVariable( "xsltready", "true" );
else setVariable( "xsltready", "false" );
</nvd:nsp>

<nvd:if test="$xsltready == 'false'">
<nvd:command name="processing-instruction.add" instruction=
"nvd-process name=&quot;xsl-transform&quot; href=&quot;index.xsl&quot;"/>
</nvd:if>

<nvd:if test="$xsltready == 'true'">
<nvd:command name="processing-instruction.add" instruction=
"xml-stylesheet href=&quot;index.xsl&quot; type=&quot;text/xsl&quot;"/>
</nvd:if>
```

3.3. Utilisation des variables

Une variable est affectée à l'aide de l'instruction "nvd:variable" et l'attribut "set" qui défini le nom de la variable. Par exemple pour affecter la valeur 2010 à la variable 'var' l'instruction suivante est nécessaire :

```
<nvd:variable set="var">2010</nvd:variable>
```

La lecture d'une variable peut s'effectuer de plusieurs façons différentes :

- A l'aide de l'instruction "nvd:variable" et de l'attribut "select" pour indiquer la variable à sélectionner :

  ```
  <nvd:variable select="var"/>
  ```

  L'élément sera remplacé par la valeur de la variable (à savoir 2010).

- Dans l'attribut d'un élément du namespace "http://www.nvdcms.org/cms/" à l'aide de la syntaxe "{$...}" ou "..." représente le nom de la variable :

  ```
  <nvd:instance id="{$var}">...</nvd:instance>
  ```

  ```
  <nvd:instance search="linux {$var}">...</nvd:instance>
  ```

  Le contenu de la variable est directement remplacée dans l'attribut ce qui nous donne.

  ```
  <nvd:instance id="2010">...</nvd:instance>
  ```

  A noter que la valeur est littérelament remplacée avec les eventuels retours à la ligne qui pourrait être contenu dans une chaine de caractères. Or il faut garder en mémoire que les retours à la ligne dans un attribut ne sont pas pris en compte dans XML.

- Dans l'attribut d'un test condition nvd:if où il est possible de mentionner une variable dans l'attibut "test" à l'aide de la syntaxe "$...", où le '...' répresente le nom de la variable :

  ```
  <nvd:if test="$var == 2010">...</nvd:if>
  ```

  A noter qu'il est toujours possible d'utiliser la syntaxe précédente (cette dernière étant prioritaire)

  ```
  <nvd:if test="{$var} == 2010">...</nvd:if>
  ```

Si pour une raison quelconque il est nécessaire d'affecter une variable à un attribut d'un élément qui n'appartient pas au namespace "http://www.nvdcms.org/cms/", il faut alors utiliser l'instruction "nvd:attribute". "nvd:attribute" fonctionne avec l'attribut "name" qui indique le nom de l'attribut à affecter à l'élément parent.

```
<element>
<nvd:attribute name="attrib">
<nvd:variable exec-priority="1" select="var"/>
</nvd:attribute>
</element>
```

Ce qui nous donne après traitement :

```
<element attrib="2010"/>
```

A noter l'ordre d'exécution *exec-priority="1"* sur l'instruction "nvd:variable" (voir Section 3.2.2). Dans le cas où cet ordre n'avait pas été précisé, l'instruction "nvd:attribute" aurait été exécutée avant "nvd:variable" (dans l'ordre d'apparation dans le document) et donc avant que la valeur '2010' n'ait été positionnée.

3.3.1. Listes

Il est possible de lire des variables sous forme de liste (en revanche lors d'une affectation à l'aide d'un set la variable est automatiquement traité comme une Map).

La lecture d'un des éléments est effectuée en mentionnant l'indice entre crochets :

```
<nvd:variable select="liste[2]"/>
```

Pour dérouler le contenu de la liste il suffit de formuler l'instruction avec des sous-éléments "nvd:value-of".

```
<nvd:variable select="liste" limit="10">
<entry>
<value><nvd:value-of select="liste"/></value>
</entry>
</nvd:variable>
```

L'élément 'nvd:value-of select="key"' est remplacé par l'indice dans la liste, alors ce que 'nvd:value-of select="liste"' sera remplacé par la valeur dans la liste à l'indice donné.

Comme nous le montrons dans la partie consacrée aux commandes (voir Section 3.5.2), les listes de variables sont également utilisées pour stocker les messages d'erreur qui ont eu lieu lors de l'exécution d'une commande.

3.3.2. Maps

Il est possible de manipuler des 'maps' de variables. Lors d'une affectation, il suffit d'indiquer entre crochets "[]" la clé associée à la variable actuelle pour stocker la valeur. Par exemple pour affecter à la variable "mapvar" les valeurs 1, 2, 3 associées respectivement aux clés A, B, C, il suffit d'ecrire :

```
<nvd:variable set="mapvar[A]">1</nvd:variable>
<nvd:variable set="mapvar[B]">2</nvd:variable>
<nvd:variable set="mapvar[C]">3</nvd:variable>
```

La lecture d'un des élements de la 'map' est effectuée en donnant la clé entre crochet :

```
<nvd:variable select="mapvar[B]"/>
```

Tout comme la liste, il est possible de dérouler le contenu d'une 'map' :

```
<nvd:variable select="mapvar" limit="10">
<entry>
<key><nvd:value-of select="key"/></key>
<value><nvd:value-of select="mapvar"/></value>
</entry>
</nvd:variable>
```

Ici, l'élément 'nvd:value-of select="key"' est remplacé par la clé, alors ce que 'nvd:value-of select="mapvar"' sera remplacé par la valeur.

3.3.3. Les variables prédéfinies

Certaines variables sont prédéfinies en fonction du contexte d'exécution. En fonction du type de variable, un préfixe spécifique est utilisé. Par exemple, les champs d'un formulaire posté en HTTP GET ou POST sont récupérables à l'aide d'un type de variables ayant pour préfixe "parameter.".

Les différents types de préfixe disponibles sont :

Préfixe	**Description**
user	Valeurs concernant l'utilisateur actuellement 'loggé'.
privilege	Valeur booléenne indiquant si l'utilisateur bénéficie du privilege mentionné en suffixe.
site	Données du site.
system	Valeurs système.
request	Valeur de la requête HTTP actuelle.
parameter	Contenus des champs soumis à l'aide d'un formulaire. Il est nécessaire de donner en suffixe le nom du paramètre 'input' concerné. La valeur récupérée peut être une liste (si l''input' était soumis plusiseurs fois avec le même nom).
soap	Contenus d'une requete SOAP. Le suffixe doit être le nom de la méthode appelée. La valeur récupérée est une 'map' ordonné avec comme clé, le nom des paramètres de la méthode.
cookie	Valeur d'un cookie dont le nom est donné en suffixe.
session	Variable de session, associé à la session en cours (l'utilisateur doit avoir un cookie de session positionné). Trois types de variables de session sont possibles : session.*, session.domain.* et session.global.* (ou * est le nom de la variable

Préfixe	Description
	conserné), avec respectivement, une visibilité sur le site en cours, sur le domaine, et sur tous les sites de la plateforme.

Par exemple, pour récupérer une valeur postée dans un formulaire web avec comme 'input name' 'message' les instructions suivantes sont à effectuer :

```
<nvd:variable select="parameter.message"/>
<nvd:if test="$parameter.message == 'Hello World'">...</nvd:if>
...
<nvd:value-of select="{$parameter.message}"/>
...
```

Avec le formulaire HTML correspondant :

```
<form method="post" action="?">
<input type="text" name="message"/>
<input type="submit"/>
</form>
```

3.4. Tests conditionnels

L'instruction "nvd:if" du langage est utilisée pour effectuer les tests conditionnels. Si le test est invalide, l'intégralité du sous arbre contenu dans l'élement est supprimé.

Il est important de bien faire attention aux priorités d'exécution lors de l'utilisation d'un test (voir Section 3.2.2).

Deux façons d'évaluer une expression sont disponibles :

- en utilisant l'attribut "test" : évaluation simple, rapide et typée de l'expression
- en utilisant l'attribut "eval" : évaluation scriptée (utilisant la syntaxe javascript) de l'expression

3.4.1. Utilisation de l'attribut "test"

L'attribut "test" permet une évalution directe et simple de l'expression. Les valeurs données sont automatiquement 'castées' dans le type le plus approprié (chaîne de caractères, nombre) pour effectuer la comparaison.

La syntaxe est simple : "(valeur1) (comparateur booléen) (valeur2)".

Les opérateurs possibles sont '==', '!=', '>', '>=', '<' et '<='. La valeur peut être une variable $... (ou '...' est le nom de la variable concerné), un nombre entier comme 2010, un nombre réel (avec '.' comme vigule) comme 20.10, une chaine de caractère comme 'Hello World' ou Hello. L'expression "null" peut être utilisée pour s'assurer qu'une variable existe. L'expression " peut être utilisé pour s'assurer qu'un variable est vide de contenu.

```
<nvd:if test="$var == Hello">...</nvd:if>
<nvd:if test="$var <= 20.10">...</nvd:if>
<nvd:if test="$var != 2010">...</nvd:if>
<nvd:if test="$parameter.var != 'Hello World'">...</nvd:if>
<nvd:if test="$parameter.var != null">...</nvd:if>
<nvd:if test="$var != ''">...</nvd:if>
<nvd:if test="10 != {$var}">...</nvd:if>
```

3.4.2. Utilisation de l'attribut "eval"

L'attribut "eval" permet une évalution scriptée de l'expression à l'aide d'une syntaxe javascript. Contrairement à l'attribut "test", la seul façon d'interagir avec des variables est d'utiliser la syntaxe {$...} (ou ... représente le nom de la variable) qui va substituer la valeur de la variable à même le script.

```
<nvd:if eval="({$parameter.var} + 5.75) == (11.5 + 6.75)">
...
```

```
</nvd:if>
```

L'avantage de l'évaluation est de permettre l'intéraction avec des méthodes javascript déclarées dans la page. (voir Section 3.7.2).

```
<nvd:script>
function value( myvar ){
return myvar+6.75;
}
</nvd:script>

<nvd:if eval="({$parameter.var} + 5.75) == value(11.5)">
...
</nvd:if>
```

Ici, l'évaluation passe par un appel à la méthode value(...) déclarée dans une instruction "nvd:script" placée, dans l'ordre d'exécution, avant le test conditionnel. La méthode se contente de retourner le résultat d'une addition.

3.5. Utilisation des commandes

3.5.1. Principe

Comme nous l'avons vu plus haut, les commandes sont des instructions qui prennent en compte des paramètres typés. Les paramètres sont passés à la commande à l'aide d'éléments "nvd:parameter".

Par exemple, dans le cas de la création de l'objet "message", nous renseignons le paramètre "name" de la commande pour préciser le nom de l'objet.

```
<nvd:command name="object.create" >
<nvd:parameter name="name">message</nvd:parameter>
<nvd:result name="id" variable="object-id" />
</nvd:command>
```

Il est possible de spécifier des paramètres d'un type particulier en renseignant l'attribut 'type' de l'élement "nvd:parameter". Les types les plus fréquents sont :

Type	**Description**
string	Chaine de caractères
integer	Entier
boolean	"true" ou "false"
date	Date (l'attribut format précise le format de la date)
float	Réel
xml	XML (pouvant être parsé à l'aide d'attribut spécifique)
soap	XML SOAP qui sera desérialisé dans le format natif

Le type "string" est pris par défaut.

Le type "date" est utilisé avec l'attribut "format" qui indique la façon dont est formatée la date (voir Annexe E. *Format de date*).

Le type "xml" est accompagné des attributs suivants :

Attribut	**Défaut**	**Description**
trim	true	Booléen qui indique si une chaine de caractères doit être 'trimmée' : supprime les 'white space' en début et fin de chaine de caractères du text node.

Attribut	Défaut	Description
parse-text-nodes	false	Booléen indiquant si le contenu des text nodes compris dans l'élement doit être parsé.
ignore-invalid-tags	false	Booléen (à utiliser avec l'attribut 'parse-text-nodes') qui rejette tous les tags malformés dans le contenu parsé.
convert-linebreaks	no	Convertit les retours à la ligne : dans le cas "html" les '\n' sont convertis en élément 'br' et dans le cas "text" les éléments 'br' sont convertis en '\n'.
max-word-length	voir fichier de configuration	Entier qui indique la taille maximum d'un mot sans césure. Si la taille est dépassée, une césure est automatiquement insérée.
allowed-tags	voir fichier de configuration	Chaine de caractères contenant la liste des éléments XML autorisés séparée par des espaces: "table tr td" autorise les éléments 'table' 'tr' et 'td' dans l'XML. Cette liste complète la liste des éléments par défaut contenu dans le fichier de configuration du logiciel.
ignore-default-allowed-tags	false	Booléen qui précise si les éléments par défaut définis en configuration doivent être pris en compte.
serialize-subtree	false	Booléen pour serialiser la sous stucture et la passe comme parametre string.
radix	10	Utilisé avec le type integer pour indiquer la base dans laquelle est écrite le chiffre.

Exemple d'utilisation :

```
<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte"
type="xml"
parse-text-nodes="false"
max-word-length="20"
convert-linebreaks="html"
allowed-tags="strong bold"
>
Ceci est un <strong>texte</strong>
très <bold>instructif</bold>
</nvd:parameter>
</nvd:command>
```

3.5.2. Gestion des erreurs

Si l'exécution d'une commande échoue, il est toujours possible de récupérer le ou les messages d'erreur associés. Il suffit de déclarer un nom de variable dans l'attribut 'error-variable' de l'élément "nvd:command". En cas d'échec, la variable contient alors une liste de messages d'erreur qu'il suffit de dérouler à l'aide de l'instruction "nvd:variable" (voir Section 3.3.1).

Un message d'erreur comprend les champs suivant : code, message et data.

Champ	Description
code	Code de l'erreur tel que définit dans la documention "Language Reference".
message	Message d'erreur type.
data	Donnée associée à l'erreur (généralement le champ concerné par l'erreur).

Ici, nous listons les erreurs (limitées à 10) de la commande "nvd:publication". Après exécution de la commande, la variable "error-var" contiendra une liste d'erreurs. La liste est déroulée à l'aide de l'instruction "nvd:variable".

```
<nvd:command name="publication.create"
publication-list-name="publicationtest" error-variable="error-var">
```

```
<nvd:parameter name="instance-id">
<nvd:variable exec-priority="1" select="instance-id"/></nvd:parameter>
<nvd:result name="id" variable="publication-id" />
</nvd:command>

<erreurs>
<nvd:variable select="error-var" limit="10">
<code><nvd:value-of select="code"/></code>
<message><nvd:value-of select="message"/></message>
<data><nvd:value-of select="data"/></data>
</nvd:variable>
</erreurs>
```

Les codes et messages d'erreur dépendent des commandes. Cependant les erreurs système suivantes sont communes à toutes les commandes :

Code	**Description**
300	Erreur interne (se référer à la console).
320	Manque un paramètre dans les attributs de la commande.
321	Paramètre malformé dans les attributs de la commande.
323	La contrainte imposée par le paramètre en attribut de la commande n'est pas respectée.
330	Manque un paramètre "nvd:parameter".
331	Paramètre du "nvd:parameter" malformé.
332	Mauvais type de paramètre "nvd:parameter".
333	La contrainte imposée n'est pas respectée dans le "nvd:parameter".
400	Privilège insuffisant pour exécuter la commande.

3.5.3. Enchaînement avec un submit HTTP en GET/POST

Comme nous l'avons présenté dans l'exemple du "Hello World" (voir Section 2.4 et Section 3.3.3), la récupération des paramètres soumis lors de la validation d'un formulaire, nécessite l'utilisation des variables 'parameter.*'.

En partant du formulaire HTML suivant :

```
<form method="post" action="commande.xml">
<input type="text" name="message"/>
<input type="submit"/>
</form>
```

La valeur postée à l'aide de *<input type="text" name="message"/>* est récupérée dans le document "commande.xml" (spécifié par l'attribut "action" dans l'élement "form"), à l'aide de la variable "parameter.message". Le suffixe "message" correspond au nom donné dans l'attribut "name" de l'élément "input" du "form".

Pour utiliser la valeur de la variable message dans une commande, il suffit de sélectionner la variable à l'intérieur d'un élément "parameter" d'une commande.

```
<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte"><nvd:variable exec-priority="1"
select="parameter.message"/></nvd:parameter>
</nvd:command>
```

A noter : l'ordre d'execution *exec-priority=="1"* au niveau de l'instruction "nvd:variable" pour que la valeur de la variable soit remplacée avant l'exécution de la commande (qui par défaut, sera alors exécutée en dernier).

3.5.4. Envoi de fichier dans un POST multipart/form-data

Dans le cas d'envoi sur le serveur de fichier par l'intermédiare d'un 'input' de type 'file', la valeur contenue dans la variable correspondante est l'URI du fichier sur le serveur.

En partant du formulaire HTML suivant :

```
<form enctype="multipart/form-data" method="post" action="commande.xml">
<input type="file" name="fichier"/>
<input type="submit"/>
</form>
```

La valeur postée à l'aide de *<input type="file" name="fichier"/>* est récupérée dans le document "commande.xml" (spécifiée par l'attribut "action" dans l'élement "form"), à l'aide de la variable "parameter.fichier". Le suffixe "fichier" correspond au nom donné dans l'attribut "name" de l'élément "input" du "form".

La valeur de la variable "parameter.fichier" est une URI qui pointe sur le fichier dans le site en cours. Le nom du fichier est le même que le fichier local. Cependant un filtre supprimant les caractères spéciaux est appliqué au nom.

Astuce

Une limitation sur la taille du fichier (définie dans la configuration de la plate-forme) limite le téléchargement. Pour des raisons de sécurité, seules certaines extensions de nom de fichiers comme .gif, .png, .jpg, .jpeg, .doc, .xls, .ppt, .pdf, .txt, .zip ou .gz sont autorisées (également définies dans la configuration). Si une autre extension est utilisé, la variable associée au fichier ne contiendra aucune valeur.

Toujours en reprenant le même exemple de commande, nous affectons au champ 'texte' de l'objet message. l'URI du fichier.

```
<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte"><nvd:variable exec-priority="1"
select="parameter.fichier"/></nvd:parameter>
</nvd:command>
```

3.5.5. Champs multiples

Dans le cas où un formulaire soumet plusieurs fois des valeurs avec le même nom d'input, il est possible de récupérer les données à travers la variable concernée qui fonctionne alors comme une liste (voir Section 3.3.1).

En partant du formulaire HTML suivant qui poste trois fois le champ 'message' :

```
<form method="post" action="commande.xml">
<input type="text" name="message"/>
<input type="text" name="message"/>
<input type="text" name="message"/>
<input type="submit"/>
</form>
```

La variable "parameter.message" est alors une liste qui comprend les 3 valeurs postées. Nous affectons en déroulant la liste de la variable, plusieurs fois le champ 'texte' de l'objet message.

```
<nvd:command name="instance.create" object-name="message">
<nvd-pre:variable select="parameter.message" limit="3">
<nvd:parameter name="texte"><nvd-pre:value-of
select="parameter.message"/></nvd:parameter>
</nvd-pre>
</nvd:command>
```

Ici, nous effectuons l'opération en 2 passes, une pour dérouler la variable et l'autre pour exécuter la commande (voir Section 3.2.3). Nous supposons que l'exécution des préfixes "nvd-pre" est effectuée avant celle des "nvd". Après le traitement du préfixe "nvd-pre" nous devrions avoir le document suivant :

```
<nvd:command name="instance.create" object-name="message">
```

```
<nvd:parameter name="texte">Texte1</nvd:parameter>
<nvd:parameter name="texte">Texte2</nvd:parameter>
<nvd:parameter name="texte">Texte3</nvd:parameter>
</nvd:command>
```

Dans le cas de la commande "instance.create" les paramètres insérés plusieurs fois sont considérés comme des champs multiples qu'il est possible de dérouler à l'aide d'un "nvd:subset" lors de l'execution de l'instruction "nvd:instance" (voir Section 3.6.2).

3.6. Publication, instances et liste de publications

La manipulation du contenu avec le logiciel NVD-CMS passe par trois domaines distincts.

- Les objets et leur champ typé.
- Les instances des objets qui représentent le contenu d'un type d'objet.
- Les listes de publications, qui contiennent des publications qui sont liées à des instances.

Une publications est un lien vers une instance, auquel est associé une date de création, de publication, et de fin de publication. Les publications sont regroupées par liste. Ainsi est il possible d'organiser des listes de publications qui organisent le contenu par thématique. Les publications disposent d'une option qui assure l'affichage où le non affichage du contenu (le flag 'hidden').

Il est possible de partager une même publication entre différentes listes. Par extension, une publication peut être utilisée dans différents listes sur plusieurs sites. Cette approche permet une organisation et un partage du contenu entre les sites.

De plus, les publications peuvent être hiérarchisées (relation père/fils entre les publications). Cette fonctionnalité permet, notamment, d'organiser les contenus sous forme d'arbre de données (cas des forums de discussion des newsgroups).

Les instances des objets peuvent avoir des champs qui pointent vers d'autres publications ou instances. Le contentu de cette instance/publication est obtenu à l'aide de instruction "nvd:subset", et ce de façon récursive.

3.6.1. Cas simple

Nous allons créer une liste de publication ayant pour nom "publicationlist", puis l'instance de notre objet "message" que nous associerons à une publication.

Création de la liste de publication :

```
<nvd:command exec-priority="1" name="publication-list.create" >
<nvd:parameter name="name">publicationlist</nvd:parameter>
<nvd:parameter name="hidden-default">false</nvd:parameter>
<nvd:parameter name="policy-exportable-default">administrator</nvd:parameter>
<nvd:parameter name="policy-modify">administrator</nvd:parameter>
<nvd:parameter name="policy-create">administrator</nvd:parameter>
<nvd:parameter name="policy-delete">administrator</nvd:parameter>
<nvd:result name="id" variable="publication-list-id" />
</nvd:command>
```

La liste de publication est créée avec les paramètres : 'hidden-default' false (par défaut toute nouvelle publication est visible), 'policy-exportable-default' administrator (par défaut toute nouvelle publication nécessitera le privilege administreur pour pouvoir être exportée), et 'policy-modify/create/delete' administrator (seul un administrateur peut modifier/créer/supprimer les publications contenues dans la liste).

Création de l'instance :

```
<nvd:command exec-priority="1" name="instance.create" object-name="message">
<nvd:parameter name="texte">Hello World</nvd:parameter>
```

```
<nvd:result name="id" variable="instance-id" />
</nvd:command>
```

Ici, nous plaçons le résultat de la création de l'instance (à savoir son id) dans la variable "instance-id" pour pouvoir l'utiliser dans la création de la publication.

Création de la publication :

```
<nvd:command exec-priority="2" name="publication.create"
publication-list-name="publicationlist">
<nvd:parameter name="instance-id"><nvd:variable
exec-priority="1" select="instance-id"/></nvd:parameter>
<nvd:result name="id" variable="publication-id" />
</nvd:command>
```

La publication est créée dans la liste de publication "publicationlist". La variable "instance-id" est utilisée pour lier la publication à l'instance précédemment créée. Par ailleurs la création de l'instance et de la liste de publication est effectuée à l'ordre d'exécution '1', alors que la création de la publication est effectuée à l'ordre d'exécution '2'. Cette opération est nécessaire pour permettre la substitution de la variable "instance-id" par l'id de l'instance créée. Bien entendu, pour l'ordre d'exécution '1', l'instruction "nvd:variable" apparait avant l'instruction "nvd:instance" dans le document.

La lecture du contenu ainsi créé dans la liste de publication s'effectue à l'aide l'instruction "nvd:publication". Le nom de la liste est donné à l'aide de l'attribut 'list-name'. Le nombre d'item récupérable est limité à 10 (inutile dans le cas de notre exemple qui ne comprend qu'une seul publication). L'ordre d'exécution est placé en '2' pour rester cohérent avec les commandes précédentes (dans le cas où toutes les opérations seraient executéées à la suite).

```
<nvd:publication exec-priority="2" list-name="publicationlist" limit="10">
<message-texte><nvd:value-of select="@texte"/></message-texte>
</nvd:publication>
```

La même instance aurait pu être placée dans plusieurs listes de publication différentes : il suffit plusieurs commandes de création de publication à l'aide de l'identifiant de l'instance concernée.

3.6.2. Champ multiple

Les instances d'objet peuvent avoir pour chaque champ, plusieurs valeurs contenues sous forme de liste. Pour affecter plusieurs valeurs au champ d'une instance, il suffit de passer plusieurs fois le même paramètre. Pour visualiser les valeurs multiples d'un champ d'une instance, il faut utiliser l'instruction "nvd:subset" sur un champ donné pour en dérouler le contenu.

Par exemple pour affecter plusieurs valeurs au champ 'texte' de l'objet 'message' nous utilisons la commande suivante :

```
<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte">Hello World</nvd:parameter>
<nvd:parameter name="texte">Hello Kitty</nvd:parameter>
<nvd:parameter name="texte">Hello Darling</nvd:parameter>
<nvd:result name="id" variable="instance-id" />
</nvd:command>
```

Les valeurs "Hello World", "Hello Kitty" et "Hello Darling" sont alors affectées au champ 'texte'.

Pour récupérer le contenu précédement créé avec les 3 valeurs comprises dans le champ 'texte' :

```
<nvd:instance id="{$instance-id}">
<nvd:subset select="texte" limit="3">
<nvd:value-of select="@texte">
</nvd:subset>
</nvd:instance>
```

A l'intérieur de l'instruction "nvd:instance", nous avons l'instruction "nvd:subset" qui séléctionne le champ 'texte' pour en lister les différentes valeurs (dans la limite de 3).

L'opération est similaire dans le cas d'une publication (liée à une instance). Par exemple :

```
<nvd:publication list-name="publicationlist" limit="10">
<nvd:subset select="texte" limit="3">
<nvd:value-of select="@texte">
</nvd:subset>
</nvd:publication>
```

3.6.3. Redirection vers une autre publication/instance

Certains champs d'instance sont redirigés vers d'autres instances. Cette approche permet d'organiser le contenu par type d'objets et évite la réécriture de données. Par exemple si nous souhaitons faire une carte de voeux avec toujours le même message mais des destinataires différents, nous allons créer une structure objet 'cartevoeux' dans laquelle nous aurons un champ de texte 'destinataire' et un champ 'messageredir' de redirection vers une instance.

La commande pour créer l'objet est la suivante :

```
<nvd:command name="object.create" >
<nvd:parameter name="name">cartevoeux</nvd:parameter>
<nvd:result name="id" variable="object-id" />
</nvd:command>

<nvd:command name="object-field.create" object-name="cartevoeux">
<nvd:parameter name="name">destinataire</nvd:parameter>
<nvd:parameter name="datatype">string</nvd:parameter>
<nvd:parameter name="minimum" type="integer">0</nvd:parameter>
<nvd:parameter name="maximum" type="integer">64</nvd:parameter>

<nvd:parameter name="policy-create">anyone</nvd:parameter>
<nvd:parameter name="policy-modify">anyone</nvd:parameter>
<nvd:parameter name="mandatory">false</nvd:parameter>
<nvd:parameter name="indexed">false</nvd:parameter>
</nvd:command>

<nvd:command name="object-field.create" object-name="cartevoeux">
<nvd:parameter name="name">messageredir</nvd:parameter>
<nvd:parameter name="datatype">redirection</nvd:parameter>

<nvd:parameter name="policy-create">anyone</nvd:parameter>
<nvd:parameter name="policy-modify">anyone</nvd:parameter>
<nvd:parameter name="mandatory">false</nvd:parameter>
<nvd:parameter name="indexed">false</nvd:parameter>
</nvd:command>
```

En partant du principe que la variable 'message-instance-id' contient le numéro d'instance de l'objet 'message' (que nous avons présenté dans les exemples précédents), nous allons créer deux instances de 'cartevoeux' dont le champ 'messageredir' est redirigé vers le message.

```
<nvd:command name="instance.create" object-name="cartevoeux">
<nvd:parameter name="destinataire">Jean Robert</nvd:parameter>
<nvd:parameter name="messageredir" type="integer">
<nvd:variable exec-priority="1" select="message-instance-id"/>
</nvd:parameter>
</nvd:command>

<nvd:command name="instance.create" object-name="cartevoeux">
<nvd:parameter name="destinataire">Pierre Paul Jacques</nvd:parameter>
<nvd:parameter name="messageredir" type="integer">
<nvd:variable exec-priority="1" select="message-instance-id"/>
</nvd:parameter>
</nvd:command>
```

Ici les deux instances de 'cartevoeux' pointent sur le même message. Pour lister le contenu des instances l'instruction "nvd:subset" est nécessaire.

```
<nvd:instance object="cartevoeux">
<destinataire><nvd:value-of select="@destinataire"/></destinataire>
<nvd:subset select="messageredir">
<message><nvd:value-of select="@texte"></message>
</nvd:subset>
</nvd:instance>
```

La redirection vers l'objet message est effectuée automatiquement à l'aide du "nvd:subset" qui sélectionne le champ redirigé 'messageredir'. A l'intérieur du "nvd:subset", les éléments "nvd:value-of" permettent de séléctionner les champs de l'objet 'message'.

Le mécanisme est le même lorsqu'il s'agit d'une publication (de la liste 'publicationlist') liée à une instance de 'cartevoeux'.

```
<nvd:publication list-name="publicationlist">
<destinataire><nvd:value-of select="@destinataire"/></destinataire>
<nvd:subset select="messageredir">
<message><nvd:value-of select="@texte"></message>
</nvd:subset>
</nvd:publication>
```

3.6.4. Affichage conditionnel d'une publication

Le contenu d'une publication peut apparaître ou non, lorsqu'elle est listée dans une 'liste de publication'. Plusieurs modificateurs conditionnent l'affichage d'une publication :

- La date de publication et la date d'expiration de la publication.
- Le modificateur (flag) 'hidden' de la publication.
- Le privilège de lecture de l'instance lié à la publication (voir Section 4.3).

Nous allons détailler les deux premiers cas de figure.

3.6.4.1. Date de publication

Lors de la création d'une publication, la date de publication est la même que la date de création, ce qui signifie qu'elle sera immédiatement visible. Il est cependant possible de spécifier une date de publication différée dans le temps. Il suffit de préciser les attributs 'publication-date' et 'publication-expire-date' lors de la création (ou la modification) de la publication pour borner la période de temps.

```
<nvd:command exec-priority="2" name="publication.create"
publication-list-name="publicationlist">
<nvd:parameter name="instance-id"><nvd:variable
exec-priority="1" select="instance-id"/></nvd:parameter>
<nvd:parameter name="publication-date"
type="date" format="dd/MM/yyyy">23/09/2002</nvd:parameter>
<nvd:parameter name="publication-expire-date"
type="date format="dd/MM/yyyy">25/09/2002</nvd:parameter>
<nvd:result name="id" variable="publication-id" />
</nvd:command>
```

Dans l'exemple précédent, la publication ne sera listée que pendant la période comprise entre le 23/09/2002 et le 25/09/2002. Le format de date utilisé est "dd/MM/yyyy" (voir Annexe E. *Format de date*).

Pour afficher une publication entre deux périodes de temps données l'instruction "nvd:publication" avec les attributs 'date-from' et 'date-to' (avec l'attribut 'format' pour spécifier le format des dates) doit être employé.

```
<nvd:publication list-name="publicationlist" limit="10"
date-from="22/09/2002" date-to="24/09/2002" format="dd/MM/yyyy">
<nvd:value-of select="@texte">
```

```
</nvd:publication>
```

Les attributs 'date-from' et 'date-to' se réfèrent uniquement à la 'publication-date' donnée lors de la création de la publication. 'publication-expire-date' n'est pas concernée par les attributs 'date-from' et 'date-to'.

L'attribut 'date-period' (à utiliser avec 'date-period-unit') couplé avec l'attribut 'date-from' et 'date-to' peut également être utilisé pour définir une période de temps. Dans le cas d'une utilisation avec 'date-from', 'date-period' définit une tranche temporelle après la 'date-from'. Dans le cas d'une utilisation avec 'date-to', 'date-period' définit une tranche temporelle après la 'date-to'.

Par exemple, pour définir une tranche temporelle qui commence le 22/09/2002 avec une durée de 12 heures :

```
<nvd:publication list-name="publicationlist" limit="10"
date-from="22/09/2002" format="dd/MM/yyyy"
date-period="12" date-period-unit="hour">
<nvd:value-of select="@texte">
</nvd:publication>
```

3.6.4.2. Publication masquée

Une publication peut être masquée afin de la rendre invisible dans une liste de publications. Cette option est utile pour définir une interface de modération de contenu qui peut alors simplement être affichée ou pas en fonction de l'état du paramètre.

Lors de la création ou la modification d'une instance dans une liste de publication, le paramètre 'hidden' indique si la publication est masquée. Dans cet exemple, nous masquons la publication identifée par l'id contenu dans la variable 'publi-id'.

```
<nvd:command name="publication.modify" id="{$publi-id}">
<nvd:parameter name="hidden">true</nvd:parameter>
</nvd:command>
```

Il est néanmoins possible de visualiser les publications en incluant celle qui sont masquées à l'aide de l'attribut 'view-hidden' de l'instruction "nvd:publication". Si ce dernier est positionné comme 'true', alors les publications qu'elles soient masquées ou visibles, sont récupérées.

```
<nvd:publication list-name="publicationlist" limit="10"
view-hidden="true">
<nvd:value-of select="@texte">
</nvd:publication>
```

3.6.5. Arborescence de publication

Lorsqu'une publication est créée, elle peut être rattachée à une publication parent. Cette fonctionnalité permet de structurer les publications sous forme d'arbres de données. L'élément "nvd:child" est utilisé dans l'instruction "nvd:publication" pour lister tous les fils d'une publication.

Pour rattacher une publication à un parent il suffit de renseigner le paramètre "parent-id" avec l'id de la publication concernée (55 dans notre exemple). Nous créons 2 publications, qui englobent respectivement les instances 200 et 201, ayant comme parent la publication 55.

```
<nvd:command name="publication.create"
publication-list-name="publicationlist">
<nvd:parameter name="instance-id">200</nvd:parameter>
<nvd:parameter name="parent-id">55</nvd:parameter>
</nvd:command>

<nvd:command name="publication.create"
publication-list-name="publicationlist">
<nvd:parameter name="instance-id">201</nvd:parameter>
<nvd:parameter name="parent-id">55</nvd:parameter>
</nvd:command>
```

Pour recupérer le contenu de la publication 55 et ses fils, nous utilisons l'instruction "nvd:child" (avec une limite de 2).

```
<nvd:publication list-name="publicationlist" limit="10">
<nvd:value-of select="@texte">
<nvd:child limit="2">
<nvd:value-of select="@texte">
</nvd:child>
</nvd:publication>
```

Lorsqu'une liste de publication est mentionnée à l'aide de l'attribut 'list-name' dans l'instruction "nvd:publication", seules les publications qui n'ont pas de parent sont listées. "nvd:child" est necéssaire pour obtenir les fils de chaque publication n'ayant pas de parent.

Pour lister une arborescence, l'attribut 'recurse' défini le nombre de fois que l'élément "nvd:child" doit être cloné dans un élément "nvd:child-recurse".

```
<nvd:publication list-name="publicationlist" limit="10">
<nvd:value-of select="@texte">
<nvd:child limit="2" recurse="2">
<nvd:value-of select="@texte">
<nvd:child-recurse/>
</nvd:child>
</nvd:publication>
```

Ici, l'élément "nvd:child-recurse" est remplacé deux fois par un clone de l'élément "nvd:child" ce, de façon recursive. Ce qui est équivalent à :

```
<nvd:publication list-name="publicationlist" limit="10">
<nvd:value-of select="@texte">
<nvd:child limit="2">
<nvd:value-of select="@texte">
<nvd:child limit="2">
<nvd:value-of select="@texte">
<nvd:child limit="2">
<nvd:value-of select="@texte">
</nvd:child>
</nvd:child>
</nvd:child>
</nvd:publication>
```

3.6.6. Moteur de recherche interne

Le contenu de chaque champ d'une instance peut être indéxé par mot clé dans le moteur de recherche interne. Lors de la création d'un champs d'objet, il suffit de mentionner que celui-ci doit être indexé. Par la suite, il est alors possible de récupérer les instances ou les publications qui remplissent les critères de recherche.

Pour spécifier un champ indexé, lors de la création du champ il suffit de donner la valeur booléenne true au paramètre 'indexed'.

```
<nvd:command name="object-field.create" object-name="message">
<nvd:parameter name="name">texte</nvd:parameter>
<nvd:parameter name="datatype">string</nvd:parameter>
<nvd:parameter name="minimum" type="integer">0</nvd:parameter>
<nvd:parameter name="maximum" type="integer">1024</nvd:parameter>

<nvd:parameter name="policy-create">anyone</nvd:parameter>
<nvd:parameter name="policy-modify">anyone</nvd:parameter>
<nvd:parameter name="mandatory">false</nvd:parameter>
<nvd:parameter name="indexed">true</nvd:parameter>
</nvd:command>
```

Toutes les instances créées de l'objet 'message' auront leur champ 'texte' indéxé dans le moteur.

Astuce

L'indexation du contenu a lieu à la création ou la modification de l'instance. Autrement dit, si le mode indéxation est declenché sur le champ d'un objet ayant déjà des instances, le contenu déjà créé reste non-indexé (seul les nouvelles instances seront indexées). Pour réindexer du contenu inexistant, il faut resaisir (modifier) le contenu.

L'indéxation décompose un contenu en mots clés. Les mots sont depourvus d'accents et de majuscules. Les nombres sont indéxés en tant que valeur numérique. Les nombres réels (ou '.' réprésente la virgule) sont également indéxés en valeur numérique. Les valeurs numériques et les chaînes de caractères représentant des nombres n'ont pas la même interprétation lorsqu'on leur applique des comparateurs booléens. Par exemple l'expression "200 < 21" n'a pas le même resultat si '200' et '21' sont des nombres ou chaines de caractères. C'est pourquoi l'indéxation tente d'identifier automatiquement des valeurs numériques.

La séléction de publications ou d'instances à l'aide du moteur de recherche est effectuée à l'aide de l'expression donnée dans l'attribut 'search'. Par exemple :

```
<nvd:publication list-name="publicationlist" limit="10"
search="hello">...</nvd:publication>
```

Ici la recherche est effectué sur tous les champs indéxés des instances de l'objet.

Pour préciser une recherche sur un champ précis, il faut indiquer ce dernier dans la requête :

```
... search="message=hello" ...
```

Plusieurs mots clés peuvent être donnés :

```
... search="message=hello message=world" ...
```

Les opérateurs OR ou AND entre les mots clés aident à affiner la recherche (AND est l'opérateur par défaut).

```
... search="message=hello OR message=world" ...
```

Les opérateurs = >, >=, <, <= peuvent également être utilisés avec le nom du champ pour élargir les possibilités de recherche. Par exemple pour obtenir des publications/instances avec un message qui contient un nombre supérieur à 10 et un nombre inférieur à 20 :

```
... search="message>10 AND message<20" ...
```

Dans le cas de recherche sur une chaine de caractères, si le caractère '*' est ajouté en fin de mot, cela indique une recherche sur tous les mots commencant par l'expression. Par exemple, pour obtenir toutes les instances qui contiennent un ou des mots commençant par 'worl' :

```
... search="worl*" ...
```

3.7. Langages de script

Les langages dans les documents XML simplifient certaines opérations et ouvrent certaines possibilités fonctionnelles intéressantes. Le logiciel NVDCMS supporte deux types de langages de script.

- NSP (NVD Script Page) : langage java compilé à la première lecture de document qui offre la possibilité de générer des noeuds XML à même le document.
- Javascript : langage javascript interprété (donc plus lent que NSP), qui offre une souplesse et une simplicité de programmation.

Les deux langages communiquent avec les variables du document à l'aide de méthodes qui leur sont propres.

3.7.1. NSP

Le code NSP est compilé à la première lecture (ou à la moindre modification). L'instruction "nvd:nsp" est utilisé pour définir le morceau de code à exécuter. Chaque code est compilé dans une classe java. Si un code identique se trouvent sur plusieurs pages, la même classe est utilisée.

Plusieurs méthodes sont disponibles pour communiquer avec les variables du document :

Méthode	**Description**
long getVariableAsLong(String varName);	Récupère la variable en paramètre comme un 'long'.
String getVariableAsString(String varName);	Récupère la variable en paramètre comme une 'String'.
Object getVariable(String varName);	Récupère la variable en paramètre comme une 'Object' (la classe dépend du type natif de la variable).
void setVariable(String varName, long value);	Affecte la valeur de type 'long' à la variable 'varName'.
void setVariable(String varName, Object value);	Affecte la valeur de type 'Object' à la variable 'varName'.

Exemple de code NSP :

```
<nvd:nsp>
long myvar = getVariableAsLong( "myvar" );
myvar += 10;
setVariable( "myvar", myvar );
return new Long( myvar );
</nvd:nsp>
```

Ici la variable 'myvar' du document est récupérée en tant que 'long'. Le contenu est incrémenté de 10 pour être remplacé dans la variable 'myvar'.

Pour des raisons de sécurité, la liste des classes autorisées (en l'occurence, dans l'exemple ci dessus : Long) est définie dans le fichier de configuration.

NSP permet également de générer des noeuds XML à l'aide de code java. A l'aide l'élement d'echappement "xml-code" qui garde sa sous-structure XML telle quelle, il est possible de générer de l'XML. Par exemple, si nous souhaitons générer 10 fois la structure avec les éléments "b" "c" suivants :

```
<b att1="1" att2="2">
<c att3="3">My text</c>
</b>
```

Il suffit d'englober le sous arbre ci dessus dans un élément "xml-code" et de boucler 10 fois à l'aide d'un code NSP (ici, l'élément "nsp-code" est également utilisé définir une zone de code NSP) :

```
<nvd:nsp>
String myValue = "Hello";
<![CDATA[ for( int i = 0; i<10; i++){ ]]>
<xml-code>
<b att1="1" att2="2">
<c att3="3">My text <nsp-code>addTextNode( myValue );</nsp-code></c>
</b>
</xml-code>
}
</nvd:nsp>
```

3.7.2. Javascript

Le code javascript est interprété lorsqu'il est placé dans un élément "nvd:script". Tout comme le code NSP, il est préférable de déclarer le code dans un noeud CDATA afin de ne pas être géné par des caractères spéciaux. Le

code javascript peut intéragir avec le document à l'aide des variables, et notamment avec un appel de méthode lors de l'évaluation d'un "nvd:if" (voir Section 3.4.2).

L'intéraction par les variables passe par l'utilisation des méthodes javascript suivantes :

Méthode	**Description**
mybean = bsf.lookupBean("variable");	Récupère l'objet qui fera le pont avec le document.
var a = mybean.get("myvar");	Donne le contenu de la variable 'myvar' du document.
var b = mybean.getLong("myvar");	Donne le contenu en 'long' de la variable 'myvar' du document.
mybean.set("myvar", value);	Affecte la valeur 'value' dans la variable 'myvar' du document.

La méthode *bsf.lookupBean("variable");* renvoie un instance d'un objet javascript qui implémente les méthodes get(...), getLong(...) et set(...,...) qui assurent le pont avec les variables du document.

Par exemple pour récupérer la valeur de la variable 'myvar' dans le document et l'incrémenter de 10 :

```
<nvd:script>
<![CDATA[
mybean = bsf.lookupBean( "variable" );
var a = mybean.getLong( "myvar" );
a += 10;
mybean.set( "myvar", a );
]]>
</nvd:script>
```

Il est possible de déclarer des fonctions qui seront appelées à travers le code javascript (notamment dans une évalution "nvd:if").

```
<nvd:script>
<![CDATA[
function value( myvar ){
return myvar+6.75;
}
]]>
</nvd:script>
<nvd:if eval="5.75 == value({$parameter.var})">...</nvd:if>
```

3.8. Transactions

Pour enchainer des instructions de façon "atomique" le développeur à la possibilité de definir des transactions. Dans une transaction, soit, toutes les opérations sont effectuées soit aucunes (tout ou rien). Si l'une des actions échoue, l'intégralité des actions qui étaient engagées dans la transaction sont annulées.

Seul les commandes marquées comme étant transactionelles (voir le document "Novadeck CMS 2.0 - Langage de référence") supportent un fonctionnement "atomique".

Une transaction est déclarée dans le "processing instruction" 'document-process' à l'aide de l'attribut 'transaction'.

```
<?nvd-process name="document-process" prefix="nvd-tx" transaction="tx1"?>
```

Une même transaction peut être associée à plusieurs préfixes de namespace.

```
<?nvd-process name="document-process" prefix="nvd-tx1" transaction="tx1"?>
<?nvd-process name="document-process" prefix="nvd-tx2" transaction="tx1"?>
```

Toute les commandes éxécutées avec le préfixe défini comme transactionnel font partie de la transaction.

```
<?nvd-process name="document-process" prefix="nvd-tx1" transaction="t1" ?>
<?nvd-process name="document-process" prefix="nvd-tx2" transaction="t1" ?>
```

```
<page xmlns:nvd-tx1="http://www.nvdcms.org/cms/"
xmlns:nvd-tx2="http://www.nvdcms.org/cms/"
>

<nvd-tx1:command name="instance.create" object-name="message">
<nvd-tx1:parameter name="texte">Hello Folks</nvd-tx1:parameter>
<nvd-tx1:result name="id" variable="instance-id" />
</nvd-tx1:command>

...

<nvd-tx2:command name="publication.create"
publication-list-name="publicationtest" error-variable="err">
<nvd-tx2:parameter name="instance-id">
<nvd-tx1:variable select="instance-id"/>
</nvd-tx2:parameter>
<nvd-tx2:result name="id" variable="publication-id" />
</nvd-tx2:command>

</page>
```

Ici, les instructions associées préfixes 'nvd-tx1' sont éxécutées avant celles du prefixe 'nvd-tx2', mais les deux font partie de la même transaction. Si la création de publication de l'instance (la commande "publication.create") échoue, alors la création de l'instance, sera annulée.

3.9. Contextes

Tout comme une transaction (voir Section 3.8) un contexte est déclaré dans le "processing instruction" ayant pour nom 'document-process'. Le contexte défini concernera l'éxécution des instruction ayant pour préfixe de namespace le nom déclaré dans le "processing instruction".

Deux types de contexte peuvent être redéfinis : le contexte du site en cours, et le contexte de l'utilisateur.

3.9.1. Contexte site

Si un contexte de sites est défini toutes les instructions du namespace seront exectutées comme sur le site désigné. L'attribut 'change-site' est utilisé dans la déclaration de processing instruction pour spécifier sur lequel les instructions seront éxécutées. Le nom de site est le nom complet site+domain ou un 'domaine parké'.

Exemple de changement de contexte de site :

```
<?nvd-process name="document-process" prefix="nvd-site" change-site="newsite.domain.com"?>
```

Ici, les instructions avec le préfixe 'nvd-site' seront considérées comme étant éxécutées sur site "newsite.domain.com".

3.9.2. Contexte utilisateur

Un context utilisateur est défini pour un namespace donné à l'aide du "processing instruction" 'document-process' Pour pouvoir executer des instructions en se faisant passer pour un autre utilisateur, il faut au préalable que les instructions d'un prefixe de namespace aient été signées à l'aide de la clé de signature de l'utilisateur concerné.

Pour signer un prefixe de namespace, la commande "security.sign-namespace" est utilisé en précisant : le fichier, le prefixe prefixe de namespace, le login de l'utilisateur (et eventuellement le domain-id de l'utilisateur si différent du domaine actuel).

```
<nvd:command name="security.sign-namespace"
uri="file.xml" namespace-prefix="nvd" login="robby">
<nvd:result name="signature" variable="mysign"/>
</nvd:command>
```

La variable de resultat "mysign" contient alors la signature (en base 64) à placer dans le "processing instruction" pour effectuer le changement de context utilisateur.

```
<?nvd-process name="document-process" prefix="nvd" change-user="robby"
change-user-signature="MCwCFDEa1xQGCH0cUeyHNy63YIfHX89zAhQPZ1LQ8+ieD/Q==" ?>
```

Les instructions du prefixe "nvd" seront alors executées comme si l'utilisateur 'robby' était loggé.

Chapitre 4. Gestion des privilèges

Le logiciel NVD-CMS intègre une gestion fine des privilèges des utilisateurs. Chaque compte utilisateur créé sur la plate-forme peut se voir attribué un (ou plusieurs) privilège qui lui donnera l'accès à un groupe de fonctionnalités. Un privilège est affecté pour un site donné. Si un utilisateur change de site, il ne bénéficie plus des privilèges qu'il avait sur le site précédent.

La plate-forme dispose d'un certain nombre de privilèges définis au niveau de la plate-forme. Avec, par ordre d'importance :

Privilège	**Description**
root	Super utilisateur qui dispose des pleins pouvoirs sur tous les sites de la plate-forme.
administator	Privilège le plus élevé disponible sur un site.
editor	Généralement attribué pour les tâche d'édition de fichier.
moderator	Généralement attribué pour la modération du contenu.
member	Généralement attribué pour la saisie de contenu et la visualiation de données privées.
owner	Cas particulier : attribué d'office à un utilisateur qui est le créateur d'un contenu.
user	Cas particulier : attribué d'office à un utilisateur identifié sur la plate-forme.
anyone	Cas particulier : attribué d'office à tout utilisateur.

Les privilèges sont organisés de façon hiérarchique. Un privilège comme "administrator" hérite des privilèges de "editor" qui lui même hérite des privilèges de "moderator" etc... Par exemple, si un accès à une donnée n'est autorisé qu'aux utilisateurs disposant du privilège "moderator", un utilisateur avec le privilège "editor" sera accrédité pour obtenir la donnée.

4.1. Gestion des droits au niveau du site

Un administrateur peut affecter des privilèges sur son site à des utilisateurs et ainsi donner l'accès à certaines rubriques d'administration. De plus un administrateur peut créer de nouveaux privilèges qui seront limités à son site.

4.1.1. Affecter un privilège à un utilisatateur

Un administrateur peut affecter (ou supprimer) un privilège à un utilisateur. Le privilège n'est valable que sur le site concerné.

Astuce

Le privilege requis pour affecter un privilège sur un site est défini en configuration. Par défaut il est fixé à 'administrator', mais il peut être modifié si nécessaire.

La commande "right.create" (et par opposition "right.remove") assure la création d'un droit sur un site (affectation d'un privilige à un utilisateur).

Par exemple, pour affecter le privilège 'moderator' à l'utilisateur 'robbykrieger' :

```
<nvd:privilege name="moderator" keep-structure="true">
<nvdpost:variable
set="privilege-id"><nvd:value-of select="id"/></nvdpost:variable>
</nvd:privilege>

<nvdpost:command name="right.create" login="robbykrieger"
user-domain-id="{$site.domain-id}" privilege-id="{$privilege-id}">
```

```
<nvdpost:result name="id" variable="right-id" />
</nvdpost:command>
```

A noter que nous effectuons un premier passage pour récupérer en variable, l'id du privilège 'moderator' à l'aide l'instruction "nvd:privilege". Puis un second passage, pour affecter le privilège à l'utilisateur.

4.1.2. Création de nouveaux privilèges

L'administrateur d'un site peut créer de nouveaux privilèges sur son site afin de gérer de façon plus fine l'accès à certaines rubriques ou données.

La commande "privilege.create" (ou "privilege.remove") est nécessaire pour la création (ou la suppression) d'un privilège sur le site. Comme les privilèges sont hiérarchisés, il est obligatoire de mentionner un 'privilege-id' parent.

```
<nvd:command name="privilege.create" parent-id="1">
<nvd:parameter name="name">caporal</nvd:parameter>
<nvd:result name="id" variable="privilege-id" />
</nvd:command>
```

4.1.3. Gestion dans le document

La variable avec le préfixe "privilege." (voir Section 3.3.3) renvoie la valeur booléenne 'true' si l'utilisateur qui exécute le document dispose du privilège mentionné en suffixe. Par exemple "privilege.moderator" contiendra 'true' sur l'utilisateur dispose du privilège modérateur.

Avec un test conditionnel, il est alors possible d'autoriser l'exécution d'un sous-arbre du document si l'utilisateur dispose du privilège adéquat. Par exemple pour limiter l'exécution d'un sous-arbre à un utilisateur disposant du privilège 'moderator' :

```
<nvd:if test="$privilege.moderator == true">
...
</nvd:if>
```

4.1.4. Système de fichiers

Le privilège requis pour l'accès aux fichiers est défini au niveau des caractéristiques d'un site. Ce privilège est notamment utilisé pour limiter l'accès à l'aide du protocol WebDAV (voir Annexe B. *Utilisation de WebDAV*) et lors de l'exécution des commandes "file.*". Le paramètre 'filesystem-privilege' de la commande "site.create" ou "site.modify" doit contenir le privilège nécessaire :

```
<nvd:command name="site.modify" id="{$site.id}" >
<nvd:parameter name="filesystem-privilege">moderator</nvd:parameter>
</nvd:command>
```

4.2. Gestion des droits au niveau d'un objet

Chaque champ d'objet spécifie le privilège requis pour créer et modifier le contenu. Par exemple, lors de la création du champ 'texte' de l'objet 'message', pour indiquer que la création du contenu du champ devra nécessiter le privilège 'anyone' et la modification 'moderator', nous exécutons la commande suivante :

```
<nvd:command name="object-field.create" object-name="message">
<nvd:parameter name="name">texte</nvd:parameter>
<nvd:parameter name="datatype">string</nvd:parameter>
<nvd:parameter name="minimum" type="integer">0</nvd:parameter>
<nvd:parameter name="maximum" type="integer">1024</nvd:parameter>

<nvd:parameter name="policy-create">anyone</nvd:parameter>
<nvd:parameter name="policy-modify">moderator</nvd:parameter>
<nvd:parameter name="mandatory">true</nvd:parameter>
<nvd:parameter name="indexed">true</nvd:parameter>
```

```
</nvd:command>
```

Les privilèges requis sont affectés pour chaque champ d'un objet. Cela permet d'avoir un objet composé de champs dont certains contenus ne sont pas forcément éditables par tous. C'est le cas d'un système de notation associé à un contenu : le contenu d'un article est créé que par un administrateur, en revanche n'importe quelle personne (anyone) peut positionner une note à un champ spécifique.

4.3. Gestion des droits au niveau d'une instance

Lors de la création ou la modification d'une instance, il est possible de définir à l'aide du paramètre 'policy-read', le privilège requis pour la lecture du contenu de l'instance. Par défaut, si aucun privilège n'est mentionné, la valeur du 'policy-read-default' de l'objet instancié est utilisé.

Si un utilisateur n'a pas le privilège adéquate pour le lecture de l'instance, celle-ci demeure inaccessible.

Par exemple pour rendre le contenu de l'objet 'message' visible uniquement pour les utilisateurs avec le privilège 'member' :

```
<nvd:command name="instance.create" object-name="message">
<nvd:parameter name="texte">Hello World</nvd:parameter>
<nvd:parameter name="policy-read">member</nvd:parameter>
</nvd:command>
```

4.4. Gestion des droits au niveau des publications

Une liste de publication définie les privilèges requis pour la création, la modification ou la suppression d'une publication dans la liste.

Par exemple pour créer une liste où l'ajout de publication nécessite le privilège 'anyone', la modification d'une publication 'moderator' et la suppression d'une publication 'editor', nous exécutons la commande suivante :

```
<nvd:command exec-priority="1" name="publication-list.create" >
<nvd:parameter name="name">publicationlist</nvd:parameter>
<nvd:parameter name="hidden-default">false</nvd:parameter>
<nvd:parameter name="policy-exportable-default">administrator</nvd:parameter>
<nvd:parameter name="policy-create">anyone</nvd:parameter>
<nvd:parameter name="policy-modify">moderator</nvd:parameter>
<nvd:parameter name="policy-delete">editor</nvd:parameter>
<nvd:result name="id" variable="publication-list-id" />
</nvd:command>
```

Au niveau même d'une publication, il est possible d'indiquer à l'aide du paramètre 'policy-exportable', le privilege requis dans le site où a été créée la publication, pour lier la publication dans une liste d'un autre site. Ce privilege est nécessaire pour empêcher n'importe quel site de pouvoir lier les publications du site sur lequel le contenu est créé.

Pour rendre une publication exportable (donner la possibilité de lier la publication sur un autre site) aux utilisateurs ayant le droit 'member' sur le site en cours :

```
<nvd:command exec-priority="2" name="publication.create"
publication-list-name="publicationlist">
<nvd:parameter name="instance-id">2010</nvd:parameter>
<nvd:parameter name="policy-exportable">member</nvd:parameter>
<nvd:result name="id" variable="publication-id" />
</nvd:command>
```

Si le paramètre 'policy-exportable' n'est pas renseigné, la valeur 'policy-exportable-default' de la liste dans laquelle est créée la publication est utilisée.

4.5. Privileges spéciaux

Les privileges 'root', 'owner', 'user', 'anyone' disposent des particuliarités détaillées ci-dessous.

Privilège	**Description**
root	Le privilège 'root' (ou super utilisateur) dispose de tous les droits sur tous les sites de la plate-forme. Ce privilege n'a pas de parent et n'est pas limité à un site donné. Ce privilege est à utiliser avec beaucoup de précaution.
owner	Ce privilege n'est utilisé qu'au niveau des champs d'un objet. Par exemple, lorsqu'un utilisateur est le créateur de l'instance d'un objet, et que la modification nécessite le privilege 'owner', seul, cet utilisateur aura le droit de mofifier le contenu. Ce privilège n'a pas de parent et n'est pas limité à un site donné.
user	Un utilisateur authentifié sur la plate-forme dispose automatiquement du privilege 'user' où qu'il soit. Ce privilège n'est pas limité à un site donné.
anyone	N'importe quel personne non authentifiée dispose du privilège 'anyone'. Ce privilège (qui n'en est pas vraiment un) est le plus faible disponible sur la plate-forme.

Chapitre 5. Multilinguisme

5.1. Gestion multilingue

Chaque instance d'objet dispose toujours d'un contenu texte associé à un 'locale'. Nous entendons par 'locale', le couple composé de la langue et du pays.

Astuce

Le 'locale' est également utilisé pour déterminer les formats de date (voir Annexe E. *Format de date*). Par exemple 'Lundi' en "fr-FR" et 'Monday' en "en-US".

Le locale est représenté sous forme d'une chaine de caractère composé de la langue en minuscule (en codage ISO-639, voir http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt) et du pays en majuscule (en codage 'ISO-3166', voir http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html) séparé par un '-'. Cette représentation est similaire à celle proposée pour les langues par la spécification XML 1.0 (voir http://www.w3.org/TR/2000/REC-xml-20001006#sec-lang-tag).

Par exemple pour le français, il existe plusieurs 'locales' comme : "fr-BE" (français Belgique), "fr-CA" (français Canada), "fr-CH" (français Suisse), "fr-FR" (français France), "fr-LU" (français Luxembourg).

Un 'locale' est uniquement associé à un contenu texte des instances. Les valeurs de type numérique ne sont pas corrélées avec un 'locale' (une valeur numérique reste la même quel que soit la langue). Par défaut le locale associé à un contenu texte est celui du site en cours. Tout site définit un timezone et un 'locale' (voir également Annexe D. *Attribut de value-of*).

Astuce

Dans le cas d'une utilisation du logiciel NVD-CMS avec le protocol HTTP, il est fortement recommandé de soummettre les données à travers un POST avec l'encoding adéquate pour permettre à la plate forme de prendre en compte proprement les caractères spéciaux de la langue concernée.

Lors de la sérialisation du document (par exemple en sortie d'un XSLT), en fonction de l'encoding sélectionné (comme UTF8 ou ISO-8859-1) certain caractère peuvent être degradé si celui-ci s'avère trop limitatif.

5.1.1. Création de contenu texte

Lors de la création d'une instance, l'attribut 'locale' peut être utilisé pour associer le contenu texte à un 'locale'. Par exemple :

```
<nvd:command name="instance.create" object-name="message" locale="en-US">
<nvd:parameter name="texte">Hello World</nvd:parameter>
<nvd:result name="id" variable="instance-id" />
</nvd:command>
```

Pour associer un contenu texte avec une autre 'locale' pour la même instance, il suffit de modifier l'instance prédédente avec le 'locale' voulu.

```
<nvd:command name="instance.modify" object-name="message" locale="fr-FR"
id="{$instance-id}">
<nvd:parameter name="texte">Bonjour Monde</nvd:parameter>
</nvd:command>
```

A ce stade la même instance dispose de deux contenus texte, un pour le 'locale' "en-US" et l'autre pour le 'locale' "fr-FR". Il est possible d'associer autant de contenus différents, qu'il existe de 'locale'.

A noter que les champs numériques ne tiennent pas compte de la 'locale'. Autrement dit lors d'une modification si un champ numérique est renseigné, la nouvelle valeur est repercutée quelque soit le 'locale' de l'instance.

5.1.2. Lecture de contenu texte

Le 'locale' du contenu texte à lire est également spécifié à l'aide de l'attribut 'locale'. En cas de défaillance (s'il y a aucun contenu texte pour le locale demandé), le 'locale' mentionné dans l'attribut 'fallback-locale' est pris. Si celui-ci est également défaillant, le locale de l'élement parent est pris. S'il n'y a plus d'élément parent disponible, le 'locale' du site est utilisé.

```
<nvd:instance object-name="message" limit="10"
locale="fr-BE" fallback-locale="fr-LU">
<nvd:value-of select="@texte" locale="fr-FR" fallback-locale="fr-CA"/>
</nvd:instance>
```

Dans l'exemple ci dessus, pour les 10 instances obtenues à l'aide de l'instruction "nvd:instance", au niveau du champ 'texte', le contenu est récupéré (s'il existe) par ordre de priorité les 'locales' en "fr-FR" puis (en cas de absence de contenu pour ce 'locale') "fr-CA" puis "fr-BE" puis "fr-LU" puis le locale du site.

Les attributs 'locale' et 'fallback-locale' fonctionnent également avec l'instruction "nvd:publication". Par exemple :

```
<nvd:publication list-name="publicationlist" limit="10"
locale="fr-BE" fallback-locale="fr-LU">
<nvd:subset select="texte" limit="3" locale="en-UK">
<nvd:value-of select="@texte" locale="fr-FR" fallback-locale="fr-CA"/>
</nvd:subset>
</nvd:publication>
```

5.1.3. Suppression de contenu

La commande "instance.remove-locale-content" permet de supprimer le contenu d'une instance donnée dans un locale donné à travers l'attribut "locale".

Si le contenu de l'instance est disponible dans un seul 'locale' et que l'attribut "locale" de la commande correspond à ce 'locale', alors l'instance est complétement supprimée (valeur numérique comprise).

Par exemple, la syntaxe de la commande pour supprimer le contenu "en-US" de l'instance 2010 est la suivante :

```
<nvd:command name="instance.remove-locale-content"
id="2010" locale="en-US" />
```

5.2. Recherche multiligue

Si un champ texte (text) d'un objet est déclaré comme étant indéxé, alors le 'locale' est pris en compte lors de l'indexation.

Pour spécifier une recherche dans un 'locale' précis, il faut associer à l'attribut "search" (voir Section 3.6.6 pour le fonctionnement de l'attribut "search") l'attribut "search-locale", qui indique le 'locale' dans lequel la recherche sera effectuée. Si aucun "search-locale" n'est mentionné, par défaut la recherche est effectuée sur les contenus indexés, quel que soit le 'locale'.

Par exemple, pour effectuer une recherche sur le mot clé 'hello' dans les publications de la liste 'publicationlist' dans le 'locale' "en-US" :

```
<nvd:publication list-name="publicationlist" limit="10"
search="hello" search-locale="en-US">
...
</nvd:publication>
```

Pour les valeurs numériques, l'attribut "search-locale" n'est pas pris en compte.

Chapitre 6. Utilisateurs et sessions

6.1. Utilisateurs

Un utilisateur dispose d'un compte dont l'identifiant est caractérisé par un login et un domaine (pris en charge par la plate-forme). Le même login peut être utilisé dans des domaines différents. Le mot de passe de l'utilisateur est chiffré et stocké en base de données. Si un utilisateur échoue plusieurs fois (en fonction de la configuration) une tentative de "login", le compte est gelé. Seul un super utilisateur peut débloquer un compte utilisateur gelé.

6.1.1. Création/Modification d'un compte

Dans pratique (et par défaut) la création d'un compte ne nécessite pas de privilège. La commande "user.create" permet de créer un compte utilisateur. Celui ci doit obligatoirement être composé d'un login, mot de passe, nom, email (qui sera utilisé pour envoyer un nouveau mot de passe si la demande en est faite).

Par défaut le domaine utilisé pour la création du compte est celui du site dans lequel la commande est exécutée.

La syntaxe des différents paramètres (login, mot de passe, nom, email) est défini à l'aide d'une expression régulière dans la configuration du logiciel. Le login ne tient pas compte des majuscules.

La commande pour créer un utilisateur est la suivante :

```
<nvd:command name="user.create" >
<nvd:parameter name="login">robbykrieger</nvd:parameter>
<nvd:parameter name="password">iloveparis</nvd:parameter>
<nvd:parameter name="email">rob@krieger.com</nvd:parameter>
<nvd:parameter name="name">Robby Krieger</nvd:parameter>
<nvd:result name="id" variable="user-id" />
</nvd:command>
```

La commande pour modifier un utilisateur est la suivante :

```
<nvd:command name="user.modify" id="{$user.id}">
<nvd:parameter name="password">iloveparis2</nvd:parameter>
<nvd:parameter name="email">rob@krieger.com</nvd:parameter>
<nvd:parameter name="name">Robby Krieger</nvd:parameter>
<nvd:parameter name="freeze">false</nvd:parameter>
</nvd:command>
```

6.1.2. Login/Logout

La procédure de login, associe un compte utilisateur à une session HTTP. Le numéro de session est déterminé à l'aide du cookie de session NVD-CMS (voir Section 6.2). Il ne peut y avoir de login sans session.

La commande "user.login" est utilisée pour logger un utilisateur. Il est possible de spécifier la durée de login à l'aide de l'attribut "duration".

```
<nvd:command name="user.login" login="robbykrieger"
password="iloveparis" duration="3600"/>
```

La commande "user.logout" est utilisée pour délogger l'utilisateur courant.

```
<nvd:command name="user.logout"/>
```

6.2. Attribution d'une session et SSO

6.2.1. Login/Logout

Le numéro de session est nécessaire pour identifier un utilisateur sur la plate-forme. Lorsqu'un utilisateur s'authentifie, le compte de l'utilisateur est associé au numéro de session.

Dans le cas des connection via le protocole HTTP, le numéro de session est stocké dans un cookie. Les navigateurs web doivent obligatoirement accepter les cookies pour pouvoir disposer d'une session.

Par défaut les documents ne forcent pas le positionnement d'un cookie de session. Le 'processing instruction' "check-session-cookie" est utilisé pour indiquer qu'il est nécessaire d'avoir un numéro de session pour exécuter correctement le document.

```
<?nvd-process name="check-session-cookie"?>
```

Le numéro de session peut être aussi bien nécessaire pour une exécution des commandes qui concerne l'utilisateur ('user.login', 'user.logout') que les variables 'user.*' ou 'privilege.*' ou encore les variables de session (voir Section 6.3).

6.2.2. Single Sign On

Quels que soient les sites hébergé sur la plate-forme, le numéro de session attribué dans le cookie de session est toujours le même. Si bien qu'un utilisateur authentifié une fois sur un site, sera reconnu sur n'importe quel site de la plate-forme.

L'utilisateur ne valide qu'une seule fois, sur n'importe quel site, son login/mot de passe : "Single Sign On". L'utilisateur est alors automatiquement reconnu, sans authentification supplémentaire, et donc sans avoir à réemettre le mot de passe à travers les requêtes HTTP.

6.3. Variable de session

Une variable de session associe le nom d'une variable au numéro de session. La porté de la variable n'est plus limité au document en cours, mais à tout document d'un site, d'un domaine, ou tous les sites.

Cependant, une variable de session est limitée dans le temps. Au bout d'une durée (fixée en configuration) d'inactivité (non lecture ou écriture d'une valeur dans la variable) celle ci est détruite.

Une variable de session permet de stocker par exemple du contenu tel qu'un caddie, ou encore des préférences temporaires.

Les variables de session fonctionne de la même façon que les variables classiques mis à part le nommage avec le préfixe 'session' qui la caractèrise. Il existe trois types de variable de session avec chacune une syntaxe et portée différente :

- session.*: la porté de la variable est limitée au site.
- session.domain.* : la portée de la variable est limitée au domaine.
- session.global.* : globale à tous les sites.

Exemple d'utilisation d'une variable de session pour un site :

```
<nvd:variable set="session.myvariable">Hello</nvd:variable>
<nvd:variable select="session.myvariable"/>
```

Exemple d'utilisation d'une variable de session sur un domaine :

```
<nvd:variable set="session.domain.domvariable">Hello Domain</nvd:variable>
<nvd:variable select="session.domain.domvariable"/>
```

Chapitre 7. Production de contenu multiterminal

7.1. Rasterisation de SVG

Il est possible de produire une image binaire à partir du document en cours, à l'aide d'un 'processing instruction'. Ce dernier doit respecter le langage SVG. Le 'processing instruction' "svg-transform" permet de produire une sortie binaire dans un format d'image (à préciser dans l'attribut "format").

Les formats d'image supportés sont : 'png', 'jpeg' et 'tiff'.

Dans le cas de 'jpeg', l'attribut "quality" (comprise entre 0.0 et 1.0) indique la qualité de compression souhaitée. Pour 'png' et 'tiff', l'attribut "force-transparent-white" indique si la couleur de fond doit être transparente (note : tous les navigateurs ne supportent pas forcément cette fonction).

Pour produire une image jpeg avec une qualité assez faible à partir du document SVG :

```
<?nvd-process name="svg-transform" format="jpeg" quality="0.2"?>
```

Pour produire une image png à partir du document SVG :

```
<?nvd-process name="svg-transform" format="png"?>
```

7.2. Utilisation de FO

Un document FO peut être transformé en fonction du format de sortie à l'aide du 'processing instruction' "fo-transform".

Les différents formats de sortie disponibles sont : 'pdf', 'rtf', 'svg', 'text', 'xml' et 'ps' (postscript).

Pour produire un PDF à partir du document FO en cours :

```
<?nvd-process name="fo-transform" format="pdf" ?>
```

Chapitre 8. Web Service

8.1. Prise en compte de requêtes SOAP

Tout comme les paramètre passés à l'aide d'une requête HTTP en GET ou POST, il est possible de récuperer les paramètres d'un appel de méthode en SOAP. Les variables avec le préfixe "soap.", permette de récupérer les paramètres de chaque méthode appelée (voir Section 3.3.3).

8.1.1. Utilisation des variables

Les paramètres d'une méthode SOAP sont récupérées à l'aide de variable "soap.". Le nom de la méthode concernée doit être indiqué pour recupérer une 'map' qui comprendra les différentes valeurs (avec le nom des paramètres en clé).

Par exemple dans un document donné, les paramètres de la méthode "setValue" sont dans la variable "soap.setValue" (en admettant que le document est exécuté suite à une requête SOAP sur la méthode setValue).

Pour que le code client (appelant la méthode distante) puisse fonctionner correctement, le document utilisé lors de l'appel doit produire en sortie un XML de réponse SOAP composé des éléments "SOAP-ENV:Envelope" et "SOAP-ENV:Body" ou "SOAP-Fault".

Pour le cas d'un appel de méthode "boolean setValue(...)" la structure type du document serait la suivante (avec la valeur booléenne "true" systèmatiquement retournée).

```
<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
<SOAP-ENV:Body>

<method:setValueResponse xmlns:method="urn:nvd"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<result xsi:type="xsd:boolean">true</result>
</method:setValueResponse>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Pour positionner une erreur il suffit de renseigner l'élément "SOAP-ENV:Fault" à l'aide d'un message d'erreur. Par exemple :

```
<SOAP-ENV:Fault>
<faultcode>SOAP-ENV:Client</faultcode>
<faultstring>Valeur incorrecte</faultstring>
</SOAP-ENV:Fault>
```

8.1.2. Etude de cas

8.1.2.1. Affectation un message

Nous allons réaliser la méthode SOAP "String setMessage(long id, String msg)" qui a pour fonction d'affecter un message 'msg' à une instance dont l'identifiant égale 'id'. La méthode retourne la valeur avant modification.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:nvd="http://www.nvdcms.org/cms/"
>
<SOAP-ENV:Body>

<nvd:if test="$soap.setMessage != null">

<!-- soap response -->
<method:setMessageResponse xmlns:method="urn:nvd"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<result xsi:type="xsd:string">
<nvd:instance id="{$soap.setMessage[id]}">
<nvd:value-of select="id"/>
</nvd:instance>
</result>
</method:setMessageResponse>

<!-- perform the command -->
<nvd:command name="instance.modify" id="{$soap.setMessage[id]}"
error-variable="err">
<nvd:parameter name="message">
<nvd:variable select="soap.setMessage[msg]" exec-priority="1"/>
</nvd:parameter>
</nvd:command>

<!-- Error if any -->
<nvd:if test="$err != null">
<SOAP-ENV:Fault>
<faultcode>SOAP-ENV:Client</faultcode>
<faultstring><nvd:variable select="err"/></faultstring>
</SOAP-ENV:Fault>
</nvd:if>

</nvd:if>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Le traitement de la rêquete se décompose en 3 parties :

- L'élement de réponse "method:setMessageResponse" qui comprend le contenu du message de l'instance sélectionnée.
- L'execution de la commande "instance.modify" qui remplace le texte du message par celui donné en paramètre.
- L'élement d'erreur qui est construit si la commande positionne une variable d'erreur.

La réponse "method:setMessageResponse" contenue dans l'élément "result" typé comme une 'String' à l'aide de l'attribut 'xsi:type="xsd:string"'.

8.1.2.2. Création d'un utilisateur

Dans l'exemple suivant, nous proposons la création un utilisateur sur la plateforme. Nous implémentons la méthode "doCreateUser" qui prend en paramètre le login, le mot de passe, l'email et le nom de l'utilisateur. La methode retourne l'id de l'utilisateur créé. En cas d'erreur une faute SOAP est retournés au client, avec la liste des messages d'erreur.

```
int doCreateUser( String login, String  password, String email, String name );
```

Le document suivant implémente la méthode doCreateUser respectant la format SOAP.

```
<?xml version="1.0" encoding="UTF-8"?>

<?nvd-process name="document-process" prefix="nvd" ?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:nvd="http://www.nvdcms.org/cms/"
>

<SOAP-ENV:Body>

<nvd:if test="$soap.doCreateUser != null">

<!-- perform the command -->
<nvd:command name="user.create" error-variable="err">
<nvd:parameter name="login"><nvd:variable
select="soap.doCreateUser[login]" exec-priority="1"/></nvd:parameter>
<nvd:parameter name="password"><nvd:variable
select="soap.doCreateUser[password]" exec-priority="1"/></nvd:parameter>
<nvd:parameter name="email"><nvd:variable
select="soap.doCreateUser[email]" exec-priority="1"/></nvd:parameter>
<nvd:parameter name="name"><nvd:variable
select="soap.doCreateUser[name]" exec-priority="1"/></nvd:parameter>
<nvd:result name="id" variable="user-id" />
</nvd:command>

<!-- IF no error return user-id -->
<nvd:if test="$err == null">
<!-- soap response -->
<method:doCreateUserResponse xmlns:method="urn:nvd"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<result xsi:type="xsd:integer"><nvd:variable select="user-id"/></result>
</method:doCreateUserResponse>
</nvd:if>

</nvd:if>

<SOAP-ENV:Fault>
<!-- If error return fault -->
<nvd:if test="$err != null">
<!-- soap response -->
<method:doCreateUserResponse xmlns:method="urn:nvd"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<faultcode>SOAP-ENV:Client</faultcode>
<faultstring>
<nvd:variable select="err" limit="100"><nvd:value-of
select="message"/> </nvd:variable>
</faultstring>

</result>
</method:doCreateUserResponse>
</nvd:if>
</SOAP-ENV:Fault>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A noter l'élément "SOAP-ENV:Fault" qui retourne en erreur au client en indiquant dans l'élement "faultstring" les différentes erreurs survenues lors de l'exécution de la commande.

8.1.3. Appel de méthode distante

A l'aide de la commande "soap.request" le développeur a la possibilité d'appeler des méthodes distantes.

L'url, la méthode et les paramètres sont fournis à la commande. Le résultat peut être récupéré en variable lorsqu'il s'agit de types connus par la plate-forme (String, Map, List etc) ou alors au format XML natif. L'attribut "soap-result" indique si tel est le cas.

Exemple de rêquete SOAP sur la méthode "mymethod" à l'url "http://myurl.com/file.xml".

```
<nvd:command name="soap.request"
url="http://myurl.com/file.xml"
targetNamespace="urn:myurn"
method="mymethod"
soap-result="false" >

<nvd:parameter name="keyword">linux</nvd:parameter>
<nvd:parameter name="limit" type="soap">
<limit xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:int">10</limit>
</nvd:parameter>

<nvd:result name="response" variable="result" />
</nvd:command>
```

Les paramètres de la commande peuvent être de type "soap" pour spécifier le type des valeurs et éviter de passer par exemple un nombre sous forme de chaîne de caractères, ce qui pourrait provoquer une erreur au niveau de la méthode distante. Le nom utilisé pour les paramètres doit correspondre au nom requis par la méthode distante.

8.2. WSDL

8.2.1. Utilisation des WSDL

Les fichier WSDL (Web Service Description Language) spécifie des services distants accessibles à travers des requêtes via le réseau.

A l'aide de la commande "webservice.request" il est possible d'effectuer une requête sur un des services décrit dans un fichier WSDL. Son fonctionnement est similaire à la commande "soap.request". L'url du fichier WSDL, le service, le port, et la méthode sont donnés respectivement dans les attributs "url", "service" "port", et "method".

Tout comme pour la commande "soap.request" les paramètres de la commande sont 'typable' avec précision lors qu'il sont de type "soap".

Par exemple, pour utiliser le WebService de Google nous utilisons la commande suivante :

```
<nvd:command name="webservice.request"
url="http://api.google.com/GoogleSearch.wsdl"
targetNamespace="urn:GoogleSearch"
service="GoogleSearchService"
port="GoogleSearchPort"
method="doGoogleSearch"
soap-result="true"
error-variable="error-var"
>

<nvd:parameter name="value">BLALBLBLBLLELLDLLLBLBLBLBLB</nvd:parameter>

<nvd:parameter name="value"><nvd:variable exec-priority="1"
select="parameter.query"/></nvd:parameter>

<nvd:parameter type="soap" name="value">
```

```
<start xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:int">0</start>
</nvd:parameter>

<nvd:parameter type="soap" name="value">
<maxResults xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:int">10</maxResults>
</nvd:parameter>

<nvd:parameter type="boolean" name="value">false</nvd:parameter>

<nvd:parameter type="soap" name="value">
<restrict xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string"></restrict>
</nvd:parameter>

<nvd:parameter type="boolean" name="value">false</nvd:parameter>

<nvd:parameter type="soap" name="value">
<restrict xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string"></restrict>
</nvd:parameter>

<nvd:parameter name="value">latin1</nvd:parameter>
<nvd:parameter name="value">latin1</nvd:parameter>

<nvd:result name="response" variable="result" />

</nvd:command>
```

8.2.2. Implémentation d'un WSDL

Dans l'exemple ci-dessous nous implémentons le fichier de WSDL qui décrit le service doCreateUser que nous avions présenté précédement (voir Section 8.1.2.2).

```
<?xml version="1.0" encoding="UTF-8"?>


<definitions name="CreateUser"
targetNamespace="urn:nvd"
xmlns:typens="urn:nvd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">

<message name="doCreateUser">
<part name="login"                  type="xsd:string"/>
<part name="password"               type="xsd:string"/>
<part name="email"                  type="xsd:string"/>
<part name="name"                   type="xsd:string"/>
</message>

<message name="doCreateUserResponse">
<part name="return"                 type="xsd:integer"/>
</message>

<!-- Port "CreateUser" -->
<portType name="CreateUserPort">
<operation name="doCreateUser">
<input message="typens:doCreateUser"/>
<output message="typens:doCreateUserResponse"/>
</operation>
```

```
</portType>


<!-- Binding -->
<binding name="CreateUserBinding" type="typens:CreateUserPort">
<soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

<operation name="doCreateUser">
<soap:operation soapAction="urn:CreateUserAction"/>
<input>
<soap:body use="encoded"
namespace="urn:CreateUser"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</input>
<output>
<soap:body use="encoded"
namespace="urn:CreateUser"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</output>
</operation>
</binding>

<!-- Service section -->
<service name="CreateUserService">
<port name="CreateUserPort" binding="typens:CreateUserBinding">
<soap:address location="http://site.domain.com/soap/soap-create-user.xml"/>
</port>
</service>

</definitions>
```

Le document se décompose en plusieurs section : service (où sont définis les services mis à disposition), binding (les différentes connections possibles au service), portType (les différentes opération/méthode et les paramètres en 'input' et 'output') et message (les données qui transitent).

Chapitre 9. Tâches planifiées

9.1. Principe

Chaque utilisateur peut définir une tâche qui sera exécutée à une date précise. L'exécution des intructions de la tâche est spécifiée un document qui est appelé par le gestionnaire qui se fait passer pour l'utilisateur sur le site dans lequel la tâche a été créée.

Lorsque la tâche est créée, l'utilisateur indique l'URL relative du document à exécuter sur le site. L'URL relative peut être accompagnée de paramètres passés en GET.

Une tâche peut être répétée plusieurs fois (1 à n) à intervalle de temps régulier.

La commande "cron.create" permet à un utilisateur de créer une tâche planifiée.

```
<nvd:command name="cron.create" >
<nvd:parameter name="relative-url">
/cron/command.xml?body=World&subject=Hello&dest=login@domain.com
</nvd:parameter>
<nvd:parameter name="execution-date"
type="date" format="dd-MM-yyyy HH:mm">23-01-2002 13:58</nvd:parameter>
<nvd:parameter name="repeat-count">10</nvd:parameter>
<nvd:parameter name="repeat-delay">60</nvd:parameter>
<nvd:result name="id" variable="cron-task-id" />
</nvd:command>
```

Dans l'exemple ci dessus, le document "/cron/command.xml" sera exécuté le 23-01-2002 à 13:58 avec les paramètres 'body=World', 'subject=Hello' et 'dest=login@domain.com'. La tâche sera ensuite, répétée 10 fois, par intervalle de 60 minutes.

9.2. Document de tâches

Un document de tâche est un fichier XML qui est interprété par la plate-forme comme n'importe quel document. Par exemple, pour réaliser une tache qui expédie un email avec un sujet et un corps passé en paramètre, nous utilisons le document suivant :

```
<?xml version='1.0' encoding='UTF-8'?>

<?nvd-process name="document-process" prefix="nvd" debug-mode="false"?>
<page xmlns:nvd="http://www.nvdcms.org/cms/">
<nvd:command name="smtp.send"  error-variable="error-var" >
<nvd:parameter name="to">
<nvd:variable exec-priority="1" select="parameter.dest"/>
</nvd:parameter>
<nvd:parameter name="from">
<nvd:variable exec-priority="1" select="user.login"/>@nvdcms.org
</nvd:parameter>
<nvd:parameter name="subject">
<nvd:variable exec-priority="1" select="parameter.subject"/>
</nvd:parameter>
<nvd:parameter name="content-type">text/plain</nvd:parameter>
<nvd:parameter name="content">
<nvd:variable exec-priority="1" select="parameter.body"/>
</nvd:parameter>
</nvd:command>
</page>
```

Si le document est utilisé avec la commande présentée dans la section précédente, un email sera exédiée à dest=login@domain.com avec comme sujet 'Hello' et comme corps 'World'.

9.3. Email d'acquittement

L'utilisateur peut recevoir un email d'acquittement afin d'être sur que la tache s'est bien executée. Il suffit d'indiquer une adresse email d'acquitement lors de la création de la tâche.

Pour indiquer l'email auquel un acquittement doit être expedié, il suffit d'ajouter à la commande "cron.create" le paramètre "ack-email". Si nous reprenons l'exemple précédent cela donne :

```
<nvd:command name="cron.create" >
<nvd:parameter name="relative-url">
/cron/command.xml?body=World&subject=Hello&dest=login@domain.com
</nvd:parameter>
<nvd:parameter name="execution-date"
type="date" format="dd-MM-yyyy HH:mm">23-01-2002 13:58</nvd:parameter>
<nvd:parameter name="repeat-count">10</nvd:parameter>
<nvd:parameter name="repeat-delay">60</nvd:parameter>
<nvd:parameter name="ack-email">robby@domain.com</nvd:parameter>
<nvd:result name="id" variable="cron-task-id" />
</nvd:command>
```

En cas de succès 'robby@domain.com' recevra un email similaire à celui ci :

```
Sujet: [Cron task acknowledge] SUCCESS.
Execution time: 215 ms.
Relative URL: /cron/task.xml?para=1
User Id: 78
Site Id: 45
```

En cas d'échec 'robby@domain.com' recevra un email :

```
Sujet: [Cron task acknowledge] FAILURE.
Error message: ...
```

Chapitre 10. Optimisations

10.1. Gestion des caches

Il est possible de définir un cache du contenu à différents niveaux: au niveau des instructions, au niveau des commandes SOAP/Webservice, au niveau des phases d'exécution du document.

10.1.1. Cache d'instruction

Au niveau des instructions de lecture qui supportent une prise en compte du cache (tel que 'instance', 'publication', 'user') il suffit d'ajouter l'attribut "cache-timeout" pour spécifier le temps en secondes durant lequel il faut conserver en mémoire le contenu récupéré.

Par exemple pour conserver 60 secondes le contenu issu d'une liste de publication :

```
<nvd:publication list-name="mylist" limit="20" cache-timeout="60">
<date><nvd:value-of select="publication-date" format="dd-MM-yyyy"/></date>
<title><nvd:value-of select="@title" parse-string="true"/></title>
<id><nvd:value-of select="id"/></id>
</nvd:publication>
```

10.1.2. Cache commandes SOAP/Webservice

Le résultat des commandes "soap.request" et "webservice.request" peuvent être cachés afin d'éviter d'effectuer systèmatiquement une requête distante à chaque exécution.

Les deux commandes prennent en paramètre l'attribut "cache-result" qui indique la durée en secondes pendant laquelle les résultats de la commande seront conservés en cache.

Exemple d'utilisation :

```
<nvd:command name="soap.request"
url="http://myurl.com/file.xml"
targetNamespace="urn:myurn"
method="mymethod"
cache-result="60"
>
...
</nvd:command>
```

10.1.3. Cache process

Au niveau d'un document, il est possible de définir un cache au niveau de chaque "processing instruction" nvd-process. Le résultat après exécution du processus est conservé en cache pour une durée en secondes donnée dans l'attribut "cache-timeout".

Si un document demandé à travers une requête dispose d'un version en cache alors l'exécution du document reprend la version extraite du cache.

Par exemple pour un document ayant les déclations de "processing instruction" suivantes :

```
<?nvd-process name="document-process" prefix="nvd"?>
<?nvd-process name="xsl-transform" href="index.xsl" cache-timeout="120" ?>
```

Ici, le résultat après transformation xsl sera caché. Si une requête sur le même document est effectué avant 120 secondes, le resultat conservé en mémoire sera retourné.

Dans un autre cas de figure, nous concervons en cache le résultat après exécution des instructions du document et effectuons systèmatiquement la transformation XSLT.

```
<?nvd-process name="document-process" prefix="nvd" cache-timeout="10"?>
<?nvd-process name="xsl-transform" href="index.xsl"?>
```

Si une requête sur le même document est effectué avant 10 secondes, le résultat conservé en mémoire sera utilisé, et le traitement du document reprendra au niveau de la transformation XSLT.

Annexe A. Type de données des objets

Les type de données des champs d'un objet sont les suivants :

Type	Description
boolean	Valeur booléene
integer	Nombre (maximum 64 bits)
string	Chaine de caractères
float	Réel de 64 bits
date	Date
redirection	Redirection vers une instance
redirection-publish	Redirection vers une publication

Astuce

Il est possible d'utiliser l'instruction "nvd:datatype" du langage pour lister tous les types de données disponibles.

Les type de données 'redirection' et 'redirection-publication' permettent de pointer respectivement, vers une instance d'un objet quelconque (d'un même site), ou une publication. Lors de la lecture des données il sera possible d'obtenir les informations de l'objet pointé à l'aide d'une instruction "nvd:subset" sur le nom du champ de type 'redirection'.

La commande "object-field.create" du langage doit être utilisée pour la création d'un champ d'objet. Par exemple :

```
<nvd:command name="object-field.create" object-name="message">
<nvd:parameter name="name">texte</nvd:parameter>
<nvd:parameter name="datatype">string</nvd:parameter>
<nvd:parameter name="minimum" type="integer">0</nvd:parameter>
<nvd:parameter name="maximum" type="integer">1024</nvd:parameter>

<nvd:parameter name="policy-create">administrator</nvd:parameter>
<nvd:parameter name="policy-modify">moderator</nvd:parameter>
<nvd:parameter name="mandatory">true</nvd:parameter>
<nvd:parameter name="indexed">true</nvd:parameter>
</nvd:command>
```

Dans cet exemple, le champ 'texte' de l'objet 'message' est créé avec un type de données char1024. Il est nécessaire de renseigner le champ pour créer une instance complète (mandatory : true). Le contenu du champ est indéxé dans le moteur de recherche interne (indexed : true). Le privilège requis pour la création du contenu dans ce champ est 'administrator', et 'moderator' pour la modification.

Annexe B. Utilisation de WebDAV

L'accès aux fichiers d'un site donné s'effectue via le protocole WebDAV (voir http://www.webdav.org). WebDAV repose sur le protocole HTTP et ne nécessite pas de configuration particulière au niveau d'un firewall (du coté du client).

Un logiciel client de transfert de fichier tel que WebDrive (http://www.webdrive.com/) ou Cadaver (http://www.webdav.org/cadaver/) supportant le protocole WebDAV est nécessaire.

Une authentification login/mot de passe est nécessaire pour se connecter à un site. Un utilisateur souhaitant accéder au fichier d'un site doit disposer du privilège adéquate (voir Section 4.1.4).

L'URL de connexion du site se présente sous la forme suivante : "http://site.domain.com/webdav/" ou 'site.domain.com' est le nom de domaine du site (il peut s'agir d'un domaine parké comme d'un sous-domaine hébergé).

Exemple d'utilisation de Cadaver dont le fonctionnement est proche du client FTP en ligne de commande.

```
robby@workstation1:~> cadaver
dav:!> open http://site.domain.com/webdav/
Authentication required for NovaDeck - Webdav Autentification on server `site.domain.com':
Username: robby
Password: ******

dav:/webdav/> ls
Listing collection `/webdav/': succeeded.
Coll:  img                                    0  Jan  1  1970
Coll:  wm                                     0  Jan  1  1970
index.xml                          6482  Dec 13 18:48
index.xsl                         48945  Dec 19 10:30

dav:/webdav/> get index.xml
Downloading `/webdav/index.xml' to index.xml:
Progress: [============================>] 100.0% of 6482 bytes succeeded.

dav:/webdav/> put index.xml index-copy.xml
Uploading index.xml to `/webdav/index-copy.xml':
Progress: [============================>] 100.0% of 6482 bytes succeeded.

dav:/webdav/> ls
Listing collection `/webdav/': succeeded.
Coll:  img                                    0  Jan  1  1970
Coll:  wm                                     0  Jan  1  1970
index-copy.xml                     6482  Dec 21 13:25
index.xml                          6482  Dec 13 18:48
index.xsl                         48945  Dec 19 10:30

dav:/webdav/> exit
Connection to `site.domain.com' closed.
```

Dans un premier temps nous nous authentifions avec le login 'robby'. Puis à l'aide de la commande 'ls' nous listons les fichiers disponibles sur le site. La commande 'get index.xml' permet de récuperer localement le fichier 'index.xml'. La commande 'put index.xml index-copy.xml' place le fichier locale 'index.xml' sur le site distant avec comme nom 'index-copy.xml'. Enfin la commande 'exit' termine l'exécution du client.

Astuce

Sur un système de fichiers UNIX, un fichier (ou répertoire) lié à l'aide d'un 'lien symbolique' avec un autre fichier provenant d'un site différent ne peut être édité que sur le site d'origine.

Annexe C. Utilisation des fichiers .htaccess

A l'aide d'un fichier ".htaccess" l'accès à un répertoire est restreint à certains utilisateurs ou des utilisateurs disposant d'un privilege sur le site.

Le fichier ".htaccess" doit être placé dans le répertoire concerné par la restriction d'accès.

Un fichier ".htaccess" se compose de plusieurs déclarations.

Nom	**Description**
AuthFormUrl	URI vers lequel est redirigé le navigateur si l'utilisateur n'est pas authentifié.
Limit	Elément qui liste les restrictions d'accès au répertoire : 'require privilege [privilege]' ou 'require user [user@domain.com]'.

Par exemple, pour limiter l'accès à un répertoire aux administrateurs d'un site et rediriger les personnes non authentifées vers l'URI "/wm/index.xml" il suffit de placer le fichier suivant dans le répertoire.

```
AuthFormUrl /wm/index.xml
<Limit>
require privilege administrator
</Limit>
```

Idem, mais en limitant l'accès aux 'moderator' et à l'utilisateur robby@domain.com et jim@domain.com.

```
AuthFormUrl /wm/index.xml
<Limit>
require privilege administrator
require user robby@domain.com
require user jim@domain.com
</Limit>
```

Annexe D. Attribut de value-of

L'élément "value-of" est utilisé dans la plupart des instructions pour replacer la valeur dans un 'textnode'. Les attributs suivants peuvent être appliqués à l'élément.

Attribut	**Description**
format	Chaine de caractères spécifiant le format d'un nombre ou d'une date. Les attributs "timezone" et "locale" peuvent être ajoutés au format pour ajuster respectivement l'heure et le formatage.
parse-string	Parse le contenu d'une 'String' en un sous-arbre XML. Le contenu doit obligatoirement être 'well-formed'.
parse-text-nodes	Parse le contenu des textnode d'XML en un sous-arbre XML. Le contenu doit obligatoirement être 'well-formed'.
url-encode	Encode le contenu de la valeur dans un format compatible avec celui des URL.
url-encoding	Indique l'encoding sous jacent de la string (UTF-8, ISO-8859-1, etc...).
timezone	Ajuste la timezone de la date qui sera affichée.
length-limit	Coupe au mot la chaine de caractères et rajoute "..." en fin de chaine.
substring-begin	Recupère une sous chaine de caractères en commencant à la position mentionée en attribut (ne fonctionne pas si 'length-limit' est utlisé).
substring-end	Recupère une sous chaine de caractères en finissant à la position mentionée en attribut (ne fonctionne pas si 'length-limit' est utlisé).
case	'upper' ou 'lower' ou 'firstupper' : modifie la casse de la chaine de caractères.
convert-linebreaks	'html' convertit les '\n' en 'br' ou 'text' qui convertit les 'br' en '\n'.
encoding	Formatte le contenu dans un encoding. Seul l'encoding "soap" est actuellement disponible et donne le contenu dans un sous arbre représentant sa valeur en SOAP. Les attributs "soap-object-name" et "encoding-style-uri" peuvent être ajoutés à l'encoding soap pour spécifier respectivement, le nom de l'objet soap et le type d'encoding soap.

Exemple de l'utilisation de l'attribut 'format' avec une date en jour/mois/année avec le locale en 'fr' le timezone ajusté à 'GMT+14:00'.

```
<value-of select="myfield" format="dd/MM/yyyy"
locale="fr" timezone="GMT+14:00"/>
```

Exemple de l'utilisation de l'attribut 'format' avec un nombre en x.xE-n.

```
<value-of select="myfield" format="0.###E0"/>
```

Exemple de l'utilisation de l'attribut 'parse-string' pour générer le sous-arbre XML.

```
<value-of select="myfield" parse-string"true"/>
```

Exemple de l'utilisation de l'attribut 'encoding' pour générer le sous-arbre XML de la valeur en SOAP.

```
<value-of select="myfield" encoding="soap"/>
```

Annexe E. Format de date

L'attribut 'format' dans le cas d'une date, permet de spécifier la façon dont faut formatter le contenu de la date.

```
<nvd:publication list-name="principale">
<nvd:value-of select="@date" format="dd/MM/yyyy"/>
</nvd:publication>
```

Dans cette exemple la date sera affichée selon le format "dd/MM/yyyy" ou dd, MM, yyyy représentent respectivement le jour dans le mois, le numéro du mois, et l'année. Le résultat pourrait être une date comme celle ci : "25/12/2002".

Lors du formattage, la 'locale' est pris en compte notament pour la transcription des valeurs textuelles comme les jours de la semaine (Lundi en 'locale' fr et Monday en 'locale' en), ou le nom des mois.

Les symboles disponibles pour le formatage sont regroupés dans le tableau suivant :

Symbole	Signification	Présentation	Exemple (valeurs locales US)
G	ère de l'humanité	Texte	AD ou BC
y	année	Nombre	1996
M	mois de l'année	Texte / Nombre	July / 07
d	jour dans le mois	Nombre	10
h	heure matin/soir (1~12)	Nombre	12
K	heure matin/soir (0~11)	Nombre	0
k	heure jour (1~24)	Nombre	24
H	heure jour (0~23)	Nombre	23
m	minute heure	Nombre	35
s	seconde dans minute	Nombre	55
S	milliseconde	Nombre	978
E	jour de la semaine	Texte	Tuesday
D	jour de l'année	Nombre	189
F	jour de la semaine	Nombre	2
w	semaine de l'année	Nombre	27
W	semaine du mois	Nombre	2
a	marqueur matin/soir	Texte	PM
z	timezone	Texte	Pacific Standard Time
'	caractère d'échapement pour du texte		
''	simple guillemet		

De plus, il est à noter que :

- pour une présentation "Texte", le fait de mettre 4 fois ou plus le symbole, donne la forme longue, sinon on obtient le format court
- pour une présentation "Nombre", les nombre inférieurs au nombre de chiffre de symbole auront un complément de zéro à gauche
- pour une présentations "Texte / Nombre", répété trois fois le symbole ou plus donne la présentation en "Texte", dans les autres cas on obtient la présentation en "Nombre"

- FO : http://www.w3.org/TR/xsl/slice6.html#fo-section

Tous les symboles non compris dans les ensembles [a...z] et [A...Z] seront traités comme du texte littéral. Par exemple : les symboles comme ':', '.', ',', ' ', '#' et '@' apparaîtront tel quel.

Quelques exemples :

Format	**Resultat (valeurs locales US)**
yyyy.MM.dd G 'at' hh:mm:ss z	1996.07.10 AD at 15:08:56 PDT
EEE, MMM d, "yy	Wed, July 10, '96
h:mm a	12:08 PM
hh 'o"clock' a, zzzz	12 o'clock PM, Pacific Daylight Time
K:mm a, z	0:00 PM, PST
yyyyy.MMMMM.dd GGG hh:mm aaa	1996.July.10 AD 12:08 PM

Annexe F. Exportation d'un site

Le contenu d'un site peut entièrement être exporté dans un fichier XML de commandes NVDCMS.

La commande "site.export" renvoie dans le flux de sortie de la requête le document qui contiendra l'intégralité du contenu du site. Le document est alors utilisable pour réimporter les données dans un nouveau site.

A noter que certain champs (avec les labels précédés du caractère '*') comme les dates de création, nécessitent le privilège super-utilisateur pour pouvoir être inséré tel quel.

Au terme de l'exécution de la commande "site.export", le traitement du document en cours est arrêté.

Exemple de document demandant l'exportation du contenu du site dans lequel il est traité.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<?nvd-process name="change-output" indent="yes"?>
<?nvd-process name="document-process" prefix="nvd"?>

<page xmlns:nvd="http://www.nvdcms.org/cms/">
<nvd:command name="site.export"/>
</page>
```