

Premiers pas avec SCENARBuilder

Version du document 3.1

Date 16 novembre 2011

Editeur UTC

Rédacteurs

Stéphane Poinsart <stephane.poinsart@utc.fr>

Contributions : Julie Wojcicki <julie.wojcicki@kelis.fr>

Laëtitia Leboeuf <laetitia.leboeuf@utc.fr>

Licence

Creative Commons - Paternité - Pas d'Utilisation Commerciale 2.0 France



| | |
|---|-----------|
| Introduction..... | 5 |
| I Préparation à une formation de modélisateur..... | 6 |
| II HelloWorld..... | 7 |
| 1 Structure du modèle..... | 10 |
| 1.1. Mise en place..... | 10 |
| 1.2. Composants du modèle..... | 12 |
| 1.2.1. Définition de l'atelier..... | 12 |
| 1.2.2. Définition de la structure XML du document..... | 14 |
| 1.3. Tests de votre chaîne..... | 19 |
| 1.4. Et ensuite ?..... | 22 |
| 2 Générateur Open Document Text..... | 23 |
| 2.1. Mise en place..... | 23 |
| 2.2. Composants et paramétrages du générateur..... | 25 |
| 2.3. Stylage..... | 32 |
| 2.4. Tests et réorganisation..... | 34 |
| 3 Générateur HTML..... | 35 |
| 3.1. Mise en place..... | 36 |
| 3.2. Composants et paramétrage du générateur..... | 38 |
| 3.3. Stylage..... | 43 |
| 3.4. Tests et réorganisation..... | 44 |
| 4 Diffusion..... | 46 |
| 5 Bilan..... | 46 |
| III HelloUniverse..... | 51 |
| 1 Méthodologie..... | 52 |
| 1.1. Analyse préliminaire..... | 52 |



| | |
|---|------------|
| 1.2. Conseils..... | 57 |
| 2 Mise en place..... | 59 |
| 2.1. Structure..... | 59 |
| 2.2. Publication Web..... | 65 |
| 2.3. Publication OD..... | 71 |
| 3 Les données de type "formulaire"..... | 73 |
| 3.1. Présentation et modélisation..... | 73 |
| 3.2. Publication HTML..... | 77 |
| 3.3. Publication OD..... | 82 |
| 4 Pages de la publication..... | 82 |
| 4.1. HTML : menu et hiérarchie de pages..... | 83 |
| 4.2. ODT : Page de garde, table des matières, numérotation..... | 87 |
| 5 Les axis..... | 92 |
| 6 Stylage HTML..... | 97 |
| 6.1. Repartir de zéro..... | 98 |
| 6.2. Modification d'un style existant..... | 103 |
| 7 Stylage OpenDocument Text..... | 107 |
| 8 Stylage Interface Auteur..... | 111 |
| 9 Constructeur d'application..... | 113 |
| 10 Bilan..... | 114 |
| IV Compléments..... | 116 |
| 1 Interface générale..... | 116 |
| 2 Éditeur..... | 118 |
| Conclusion..... | 119 |
| Glossaire..... | 121 |



Introduction

Ce guide :

- Vous initie à la création de modèles documentaires Scenari, des générateurs OpenDocument Text et HTML.
- N'est pas exhaustif (la quantité de chose à apprendre est importante et évolue régulièrement).
- S'adresse aux personnes à **profils techniques**, qui maîtrisent bien les outils informatiques, le HTML / CSS, l'utilisation d'OpenOffice...



I Préparation à une formation de modélisateur

Ces tutoriels sont des pré-requis à la réalisation des formations de l'académie des modélisateurs¹.



Réglementaire

La réalisation du tutoriel Helloworld est nécessaire pour une bonne préparation aux formations SCENARBuilder.

Premier tutoriel :

> "cf HelloWorld", page 7.

Notamment les points techniques suivants :

> "cf Structure du modèle", page 9.

> "cf Générateur Open Document Text", page 22.



Complément

La réalisation du tutoriel HelloUniverse est un plus pour avancer plus vite lors des formations SCENARBuilder, mais peut très bien **être réalisé à la suite d'une formation (une ou deux semaines après)**.

Certains concepts doivent obligatoirement être acquis pour pouvoir réaliser un premier projet de modélisation. Ils sont abordés lors des formations, mais une approche tutoriel peut vous aider à les assimiler.

La conception :

> "cf Analyse préliminaire", page 52.

Les points techniques sensibles sont les suivants :

> "cf HTML : menu et hiérarchie de pages", page 83.

> "cf Les axis", page 92.

1 - <http://scenari-enterprise.com/co/academy.html>

II HelloWorld



Objectif

Cette partie vous apprendra en quelques heures à concevoir un modèle le plus simple possible (la tradition souhaite qu'un tel exemple soit appelé HelloWorld, mais appliqué aux chaînes éditoriales ce n'est pas à prendre au sens strict). En résultat, vous aurez donc un micro modèle, avec la structure suivante : un ensemble de parties, avec pour chacune un titre.

Ce modèle simple et basique n'en sera pas moins une vraie chaîne éditoriale, ce qui explique sa **complexité**. Il faut assembler certains composants indispensables à toute chaîne, mais une fois en place les modifier ou en rajouter est une opération relativement simple.

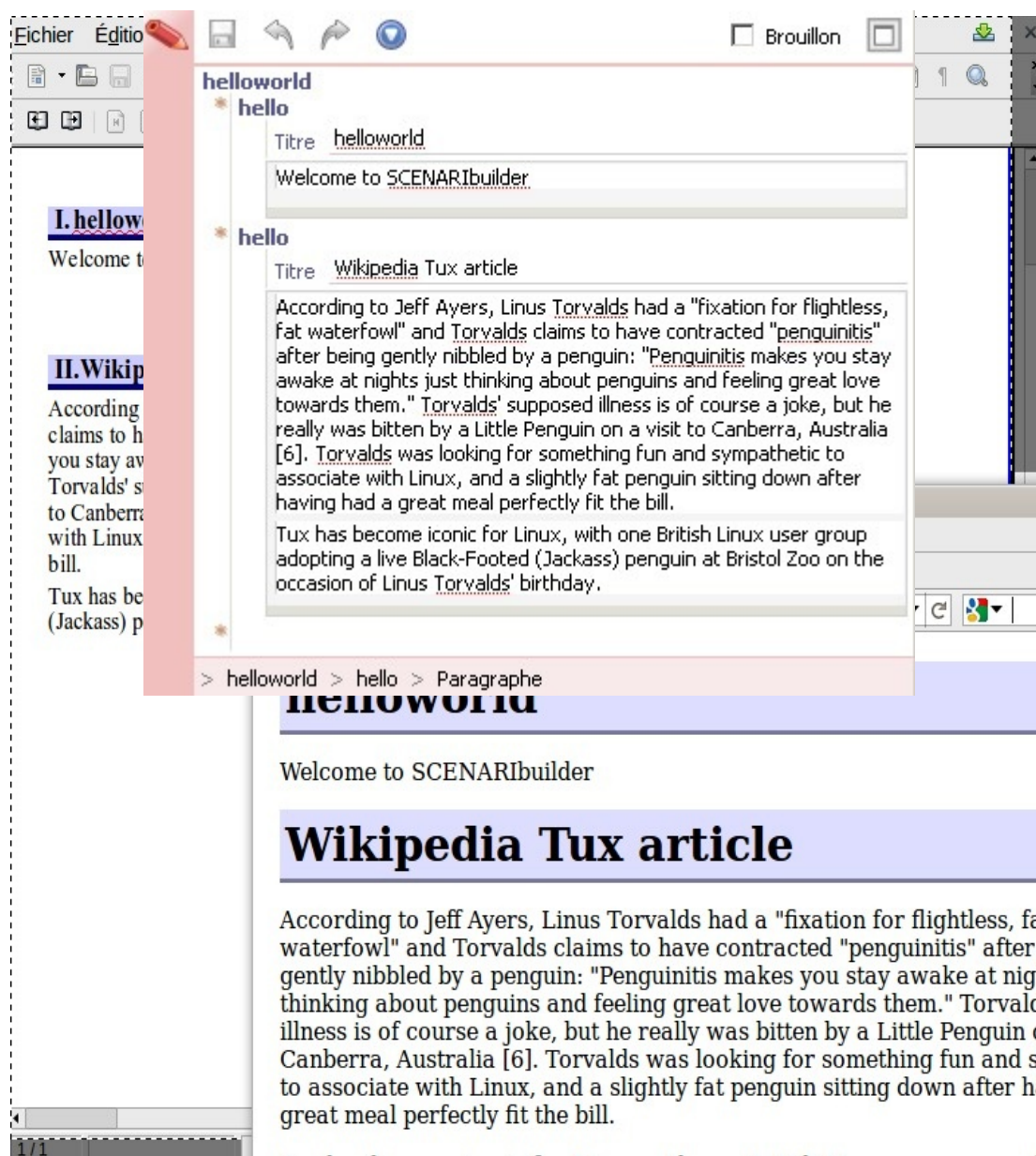
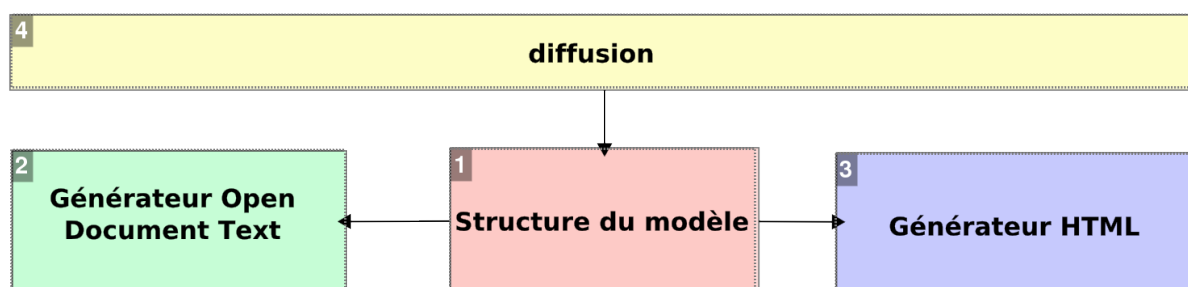


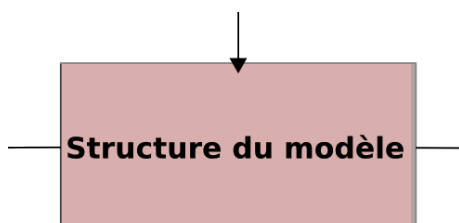
Fig. 1 Présentation du résultat souhaité

La structure du modèle HelloWorld





1-

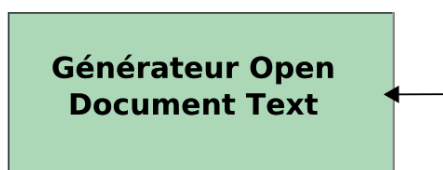


Structure du modèle

les items, les parties, les champs de données utilisables dans l'interface de l'auteur. C'est la partie indispensable à tout modèle, et par celle-là que nous allons commencer.

> "cf Structure du modèle", page 9.

2-



Générateur OpenDocument Text

la transformation du contenu en un fichier **.odt** (ou **pdf**) et son stylage.

> "cf Générateur Open Document Text", page 22.

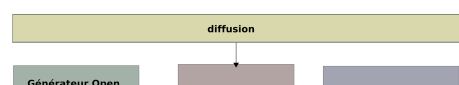
3-



Générateur HTML

la transformation du contenu en un site web et son stylage.

4-



Diffusion

créer des **wsppack** pour SCENARIchain, créer des applications autonomes, standalone*.

> "cf Diffusion", page 46.

1 Structure du modèle

1.1. Mise en place



Objectifs

Vous avez un projet "HelloWorld", techniquement la première étape dans le début de chaque projet, c'est de préparer un atelier, ce qui est décrit dans cette première partie "mise en place".



Prérequis

Vous devez avoir SCENARibuilder² et [OpenOffice.org](http://openoffice.org) installés sur votre ordinateur avant de commencer ce tutoriel.



Créer l'atelier HelloWorld

Dans SCENARibuilder, il est courant de faire un atelier par nouveau modèle ou par famille de modèles.



Créer un atelier **HelloWorld** en cliquant dans le menu sur *Ateliers* > *Ajouter un atelier...*



Définir le code de cet atelier : HelloWorld



Choisir  *Modèle des modèles*



Créer

Espace

Votre nouvel atelier est totalement vide, vous allez créer un espace* **helloworld** à l'aide du bouton Créer un espace.



Modelets

Ensembles d'items SCENARI récurrents

- **binaries** : ressources (image, vidéo, ...);
- **showPhoto** : album photos avec effets visuels.

2 - <http://scenari-platform.org/projects/scenari/fr/download/co/>



Les modelets sont des bibliothèques d'items préfabriqués, des listes de fichiers (.model ou .transf...) qui permettent à un modélisateur d'avoir des éléments prêts à être intégrés dans le modèle, ce qui est plus pratique que de commencer avec un atelier totalement vide.



Intégrer le modelet base

Préalable

Avoir téléchargé le modelet base-binaries³, sous forme d'archive ZIP.



Ouvrir les propriétés de l'atelier.

> "cf Interface générale", page 117.



Dans l'onglet **Statut**, à la ligne **Stockage des items**, cliquer sur l'icône de dossier. Cela ouvre l'espace de stockage des sources de votre atelier.



Extraire le zip dans ce répertoire de manière à avoir **binaries** et **base** en frères de **helloworld** (au même niveau)



Actualiser l'atelier

> "cf Interface générale", page 117.

Résultat

Vous obtenez la présentation suivante, dans votre atelier **HelloWord**:

- > **base**
- > **binaries**
- > **helloworld**

3 - http://scenari-platform.org/trac/modelet/attachment/wiki/WikiStart/base_binaries_3-7.zip



Les autres modelets

Pour d'autres besoins spécifiques, d'autres modelets sont disponibles sur le [wiki du projet modelet](#).

1.2. Composants du modèle

1.2.1. Définition de l'atelier



wspDef

Item permettant de définir un atelier (workspace) de SCENARichain :

- les classes d'items autorisées,
- le code par défaut de l'atelier,
- la gestion des versions
- le nom du **wspack** qui sera créé
- ...



projet.wspdef

Les générateurs



```

sm:wspDefinition keyWsp="OpaleSupStarter" name="OpaleSup "starter"" defaultWspCode=" "
majorVersion="3" mediumVersion="0" minorVersion="1"

sm:publicClasses
  sm:class sc:refUri="ue.model (/academic/model/Ue)" allowCreator=" yes"
    sm:mainView
      sm:formEditorTab name="Edition Opale" display=" visible" formEditorKey=""
      sm:blocksTab name="Publications" display=" visible"
      sm:generatorsBlock name="Publications" displayMode=" lockedOpen"
        sm:genBox
          sm:generator sc:refUri="paper.generator (/academic/gen/quadra/paper)"
          sm:deployment ...
        sm:genBox
          sm:generator sc:refUri="web.generator (/academic/gen/quadra/web)"
          sm:deployment ...

sm:wspDefinition keyWsp="showOrg" name="Présentation d'une organisation" defaultWspCode="showOrg"
majorVersion="3" mediumVersion="1" minorVersion="0"

sm:publicClasses
  sm:group ...
  sm:group name="Contenus"
    sm:group name="Site web"
      sm:class sc:refUri="wSiteML.model (/showOrg/model/webSite)" allowCreator=" yes"
        sm:mainView
          sm:formEditorTab name="Edition" display=" visible" formEditorKey=""
          sm:blocksTab name="Informations" display=" visible"
          sm:previewBlock name="Visualisation" displayMode=" openedByDefault"
          sm:technicalInfoBlock name="Technique" displayMode=" openedByDefault"
          sm:blocksTab name="Publications" display=" visible"
          sm:generatorsBlock name="Publications" displayMode=" openedByDefault"
            sm:genBox
              sm:generator sc:refUri="epure.generator (/showOrg/gen/web/skins/epure)"
            sm:genBox
              sm:generator sc:refUri="skyWeb.generator (/showOrg/gen/web/skins/skyWeb)

```

Écran 49

Le wspdef pointe vers les générateurs qui seront utilisés dans la chaîne éditoriale.



Prononciation

Au sein de la communauté Scenari, wspdef se prononce « W-S-P-def »



Créer la définition de l'atelier



Se placer sur l'espace **helloworld**



Créer un item, choisir dans la fenêtre de création d'item, *Gestion des ateliers* > *wspDefinition*



Le nommer helloworld.wspdef (nom par défaut).



Créer



Deux champs sont obligatoires dès la création de cet item : la clé du modèle et son nom. Renseigner ces champs avec **keyWsp**=helloworld et **name**=Salut le monde .



keyWsp

Sous forme de code, est l'identifiant de votre modèle documentaire au sein de la sphère des modèles Scenari.

Il est associé à un nom d'affichage, le name, qui apparaîtra pour l'auteur lorsqu'il créera un atelier, pour définir la liste des modèles documentaires qui lui seront proposés.



Version

majorVersion, **mediumVersion**, **minorVersion** : par exemple, nous créons le modèle helloworld version 0.0.1. Le changement de version peut avoir des conséquences sur les mises à jours lors de la diffusion du modèle.



publicClasses

Types d'items externes que peut créer l'auteur.



1.2.2. Définition de la structure XML du document



model

Comme dans toute application Scenari, un atelier contient des items et chaque item a : un nom de fichier, un type.

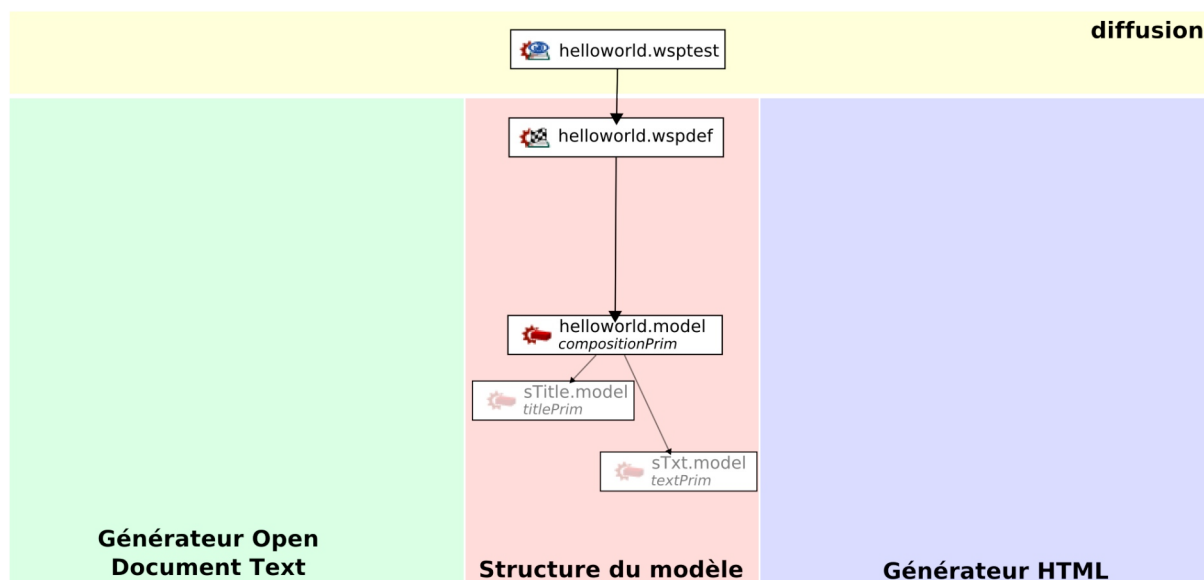
L'item de modélisation doit entre autres définir :

- **un code** : identifiant utilisé en interne par le système SCENARI. Cela doit être une suite de caractères alphabétiques, courte de préférence, où il est d'usage de le faire correspondre au nom du fichier (sauf cas particulier).
- **un name** : le nom tel que l'auteur peut le lire dans l'interface d'édition. Il peut contenir des espaces et des caractères accentués.

Cet item sert à spécifier quels champs ou paragraphes de données seront disponibles à l'intérieur de l'éditeur, de les paramétrer et de choisir comment ils seront organisés entre eux (et implicitement quel sera la structure XML des documents écrits par l'auteur lorsqu'ils sont enregistrés).

Vos items

Pour définir la structure de votre premier modèle, vous allez devoir créer 3 items, et faire appel à 2 items du répertoire modelets.





Créer helloworld.model

C'est le moment de choisir un type pour l'item racine.

Dans notre modèle, nous allons permettre à l'auteur de créer des items de type **helloworld** à partir de l'interface d'édition, nous devons donc créer dans notre atelier un item correspondant qui est listé dans les `<publicClasses>` du `wspdef`.



Se placer sur **helloworld.wspdef**



Ouvrir `sm:publicClasses` puis `sm:classe` par double clic sur ces noms de balise



Faire un clic droit sur l'item vide se situant à côté de la balise `<class>`



Choisir dans le menu contextuel *Créer un nouvel item...*



Écran 50



À retenir

Dans le reste de ce tutoriel, cette méthode sera à utiliser pour la plupart des créations d'items.



L'écran de création des items s'ouvre.



Choisir *Primitives* > *Primitives d'organisation* > *compositionPrim*.

Les compositionPrim sont utilisées pour tous les contenus organisés en parties, chapitres, blocs d'informations, sections...



Nommer l'item **helloworld.model**



Créer



Enregistrer le wspdef



Paramétrer helloworld.model

Objectif

```
sm:compositionPrim name="helloworld" category=" " info=" "
sm:help ...
sm:identification targetNamespace="scenari-platform.org:helloworld" targetPrefix="hw" code="helloworld" itemExtension="xml"
sm:structure
  sm:meta sc:refUri=" " usage="required"
  * sm:set usage="required"
    * sm:part code="hello" name="hello" family=" " internalized="always"
      * sm:meta sc:refUri="sTitle.model (/base/sTitle)" usage="optional"
      * sm:allowedModel sc:refUri="sTxt.model (/base/sTxt)" restrictUserDep=" "
    *
  *
  * sm:constraint ...
sm:authoring
  sm:icon sc:refUri=" "
  sm:itemNameCreator ...
  * sm:formEditor ...
  * sm:htmlPreview ...
  sm:previewMini ...
```

Écran 51

Nous allons reproduire la structure de cette capture d'écran.



Renseigner les champs obligatoires

name, le nom affiché dans la fenêtre de création d'items pour votre futur auteur.




namespace, prefix : En plus du code, elles sont mises en commun par un namespace, et sa version abrégée, le préfixe. Dans les cas courants, vous allez déclarer toujours le même namespace et préfixe pour tous les fichiers d'un atelier builder. L'intérêt principal de ces informations est de permettre à différents modélisateurs de partager des modèles avec le même nom. Ces noms sont utilisés en interne dans les fichiers de contenu que l'auteur va enregistrer par les éditeurs SCENARI. Le préfixe doit être court et ne pas commencer par la lettre "s" qui est réservée (s pour scenari).



Enregistrer



Créer un set de part, renseigner tout de suite le code et le name de la part.

set : Un set signifie que l'auteur pourra créer autant de parts qu'il le souhaite (dans l'éditeur, cela déclenche l'ajout d'un bouton  à gauche de la part). En l'absence de set, le document ne pourrait contenir qu'une seule partie.

part : Une "part" est un champ de contenu qui pourrait avoir une sous-structure, en faisant appel par exemple à d'autres compositionPrim.



Modifier les attributs à set : required, part internalized :always.

usage -> optional, usage -> required : Dans l'interface d'édition, l'auteur de contenu devra obligatoirement remplir toutes les parts qui ont été paramétrées par le modélisateur en tant que "required", si ce n'est pas le cas, des croix rouges apparaissent dans le contenu.

internalized : En fonction du choix du modélisateur, l'auteur pourra éventuellement, ou sera obligé de créer les parties en tant qu'item externe*. Rappel : il faut aussi que les <allowedModel> des parts soit déclarées en tant que <publicClasses> dans le **wspdef** pour que cette création d'item soit autorisée.



Définir la structure XML que devra suivre la part créée

allowedModel : C'est la liste des modèles que l'auteur va pouvoir placer à l'intérieur de ce champ. Ne créez pas un nouvel item, mais placez-y par glisser-déposer depuis le volet d'exploration l'élément "paragraphe de texte" du modelelet **/base/sTxt/sTxt.model** que vous avez installé au début de ce tutoriel.



Définir une méta pour cette part (astérisque entre `sm :part` et `sm :allowedModel`)

meta : Des méta-données associées à la `<part>`. Dans notre modèle, les parts **peuvent avoir** des titres, nous allons donc utiliser dans les modelets :
`/base/sTitle/sTitle.model`



Enregistrer



authoring

Il n'est pas nécessaire de remplir cette partie du `.model`, elle est essentiellement utilisée pour le stylage de l'éditeur.



namespace et prefix

Dans les cas courants, vous allez déclarer toujours le même namespace et préfixe pour tous les fichiers d'un atelier builder. L'intérêt principal de ces informations est de permettre à différents modélisateurs de partager des modèles avec le même nom. Ces noms sont utilisés en interne dans les fichiers de contenu que l'auteur va enregistrer par les éditeurs Scenari. Le préfixe doit être court et ne pas commencer par la lettre "s" qui est réservée (s pour scenari).

1.3. Tests de votre chaîne

Félicitation, vous avez créé votre premier modèle helloworld...



Tester la modélisation



Créer un item `wsptest` dans l'espace **helloworld**



Se placer sur cet item (l'ouvrir)



Double-cliquer sur l'icône de référence vide, au dessus de **Sélectionnez un item de type 'wspdef'**



Choisir **helloworld.wspdef** qui doit apparaître en gras dans la liste.



Cliquer sur Sélectionner

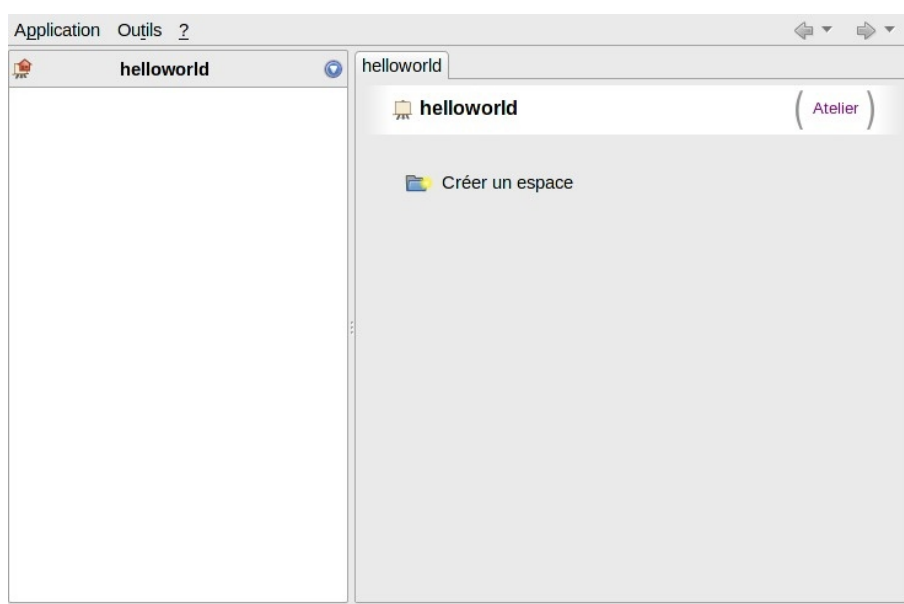


Compiler



L'interface **SCENARitest** s'ouvre.

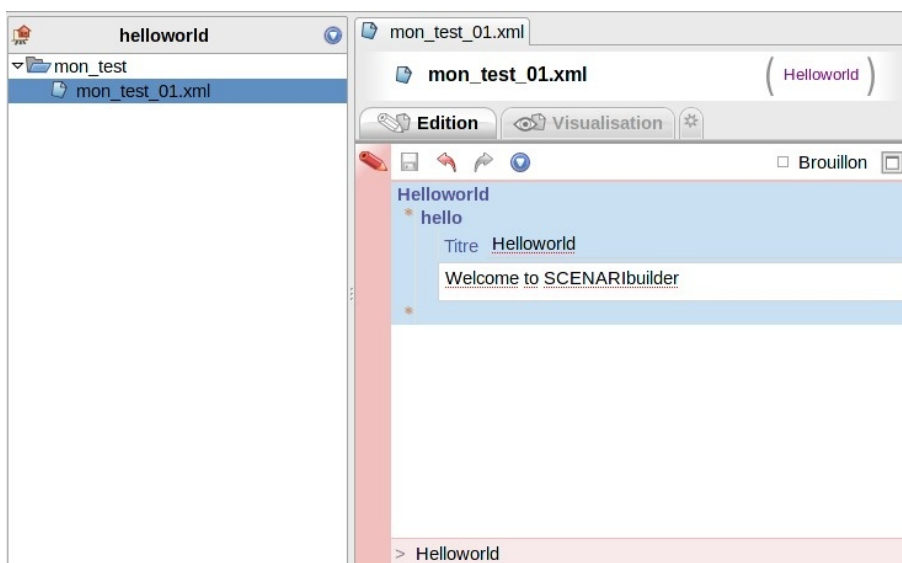
C'est une interface d'édition dédiée aux tests rapides lors du développement des modèles à partir de **SCENARibuilder**, elle ressemble fort à celle de **SCENARichain**. Vous pouvez y créer du contenu respectant les règles de votre modèle documentaire : un espace, un item helloworld, des parties...



Écran 52



Créer du contenu de test : créer un espace, un item, taper du texte, enregistrer.



Écran 53



Écran de contrôle

La vue **Contrôles** du **.wspdef** permet de contrôler les **points techniques** critiques sources de **dysfonctionnement** d'un modèle : les erreurs de modélisation, de paramétrage des publications.

En cas d'erreur, ou juste pour vérifier si tout va bien, faites appel à l'**écran de contrôle** à partir de la vue correspondant sur l'item **helloworld.wspdef**:



Écran 54



1.4. Et ensuite ?



La génération

À la fin de cette première partie, votre modèle fonctionne dans l'éditeur, mais vous n'avez aucun moyen de publier votre contenu. Nous allons maintenant poursuivre notre conception de la chaîne éditoriale helloworld par la création de générateurs dans la suite de ce document.

2 Générateur Open Document Text

2.1. Mise en place



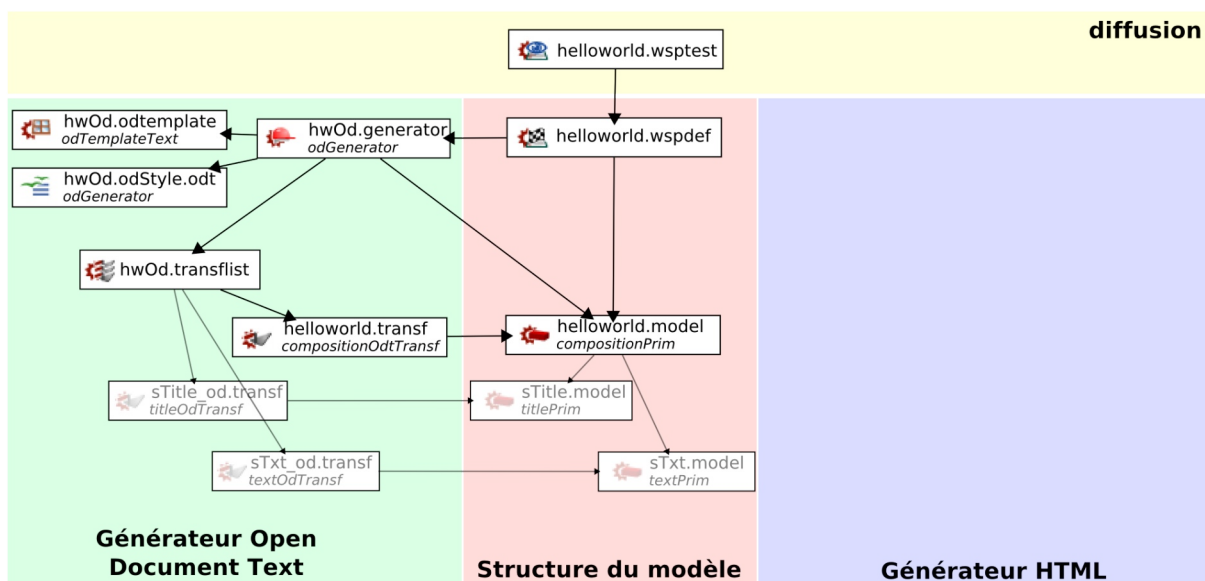
Générateurs (Les)

Si vous créez un modèle sans générateurs, les auteurs vont pouvoir écrire du contenu, mais il ne sortira jamais de sa forme "source XML". Pour transformer les données saisies, l'auteur utilise les générateurs intégrés au modèle. Vous devez créer ces générateurs dans SCENARibuilder, les lier et les adapter à la structure de votre modèle.

Schéma de fonctionnement

Dans le schéma suivant, nous allons expliquer le fonctionnement commun aux 2 générateurs du modèle helloworld.

- **Les générateurs sont déclarés dans le `wspdef`**, en ajoutant une vue de [publication](#).
- Un générateur possède un ou plusieurs **templates**, un gabarit qui agence la mise en forme globale d'une page.
- Pour chaque champ de contenu à publier, les générateurs font appel au paramétrage des **transformeurs** : les composants qui permettent le passage des données des primitives **.model** vers un support spécifique.
- Des fichiers de **stylage** interviennent aussi, pour définir couleurs, polices et images de fond.



Création du générateur



Truc & astuce

Nous allons créer le générateur OpenDocument HwOd à partir du **wspdef**



Se placer dans le **wspDef**.



Choisir sous le **sm:class** du .model **sm:mainView** puis **sm:formEditorTab** de **name Edition**



Nommer le **blocksTab** Publications, ainsi que le **genertorsBlock** qu'il contient.



```
sm:class sc:refUri="helloworld.model" access="readWrite" sortKey=""
sm:mainView
  sm:formEditorTab name="Edition" idInView="" display="visible" formEditorKey=""
  sm:blocksTab name="Publications" display="visible" idInView=""
  sm:generatorsBlock name="Publications" idInView="" displayMode="memorized"
    sm:genBox name="" idInView=""
      sm:generator sc:refUri=""
    sm:deployment...
```

Écran 55



Par clic droit sur l'élément vide au niveau de `sm:generator`, *Créer un nouvel item*.



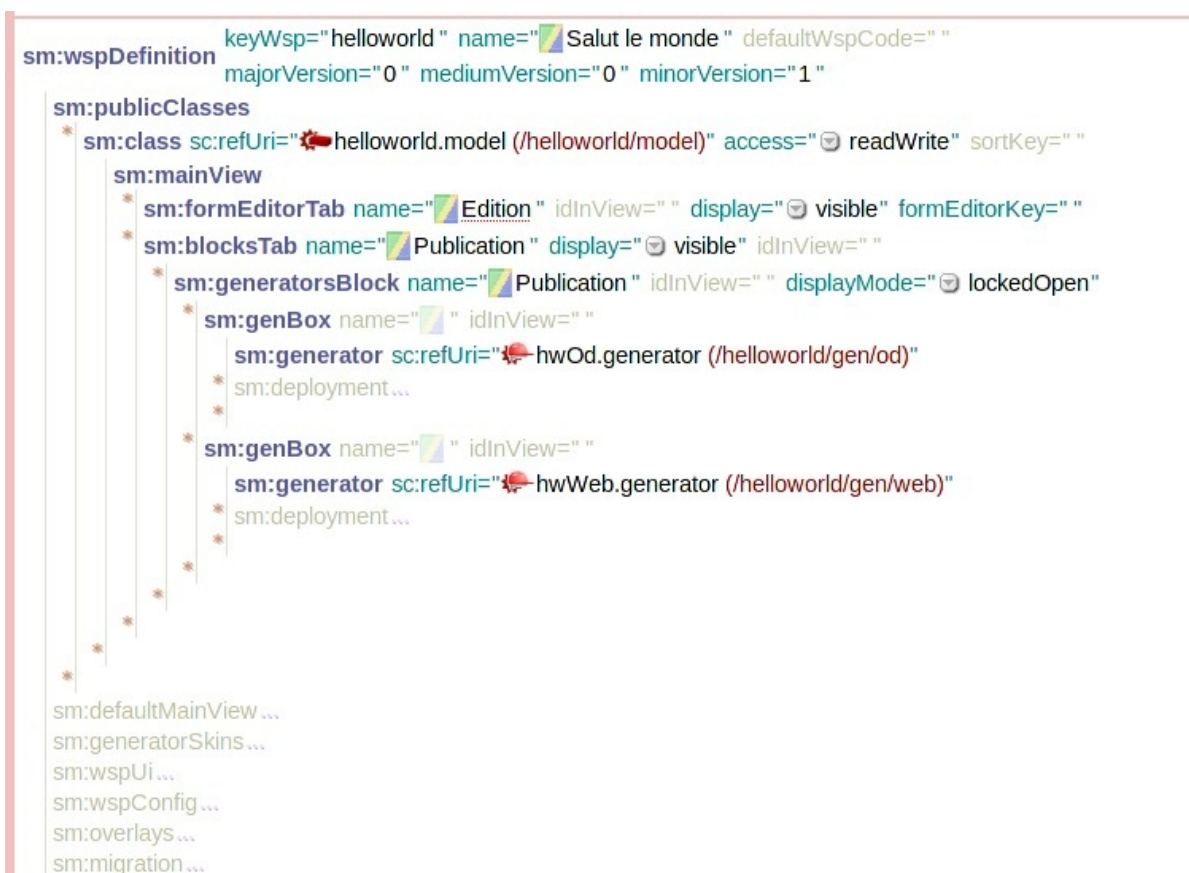
Choisir *Publication* > *Générateurs* 'openDocument' > *odGenerator*



Le nommer `hwOd` et `Créer`



Enregistrer



Écran 56

2.2. Composants et paramétrages du générateur

Générateur



Générateur

Un générateur est un ensemble d'instances de primitives permettant de spécifier les **règles de publication** d'un support dans un **format donné**.

Pour chaque entrée souhaitée dans les vues Publication, on crée en un générateur **.generator**, du type de sortie que l'on souhaite avoir : web/xhtml ou openDocument.



Vous pouvez avoir 2 publications différentes dans le format de fichier de sortie, dans le stylage, dans l'agencement des parties. Vous pouvez avoir une publication avec des informations supplémentaires (corrigés d'exercices, balises Exemple) et une sans. À chaque fois cela donne lieu à la création d'un nouveau **.generator**, qui est le point de départ de toute nouvelle publication ou déclinaison d'une publication existante.

OD et HTML

Il existe au moins un générateur par support de publication: papier et web.

Plusieurs générateurs d'un même type peuvent partager leurs styles.

Les générateurs SCORM sont des générateurs issu du web, ils utilisent en général le même dossier de styles que la publication web.



Complément

Règles de publication :

- transformer le contenu selon le canal de diffusion cible : FormeX -> FormeY
- ré-ordonner le contenu;
- sélectionner le contenu : synthèse, fiches, ...;
- définir l'IHM.

Définir le générateur

- Quel format ?
- Quelle structure du support final ?
- Quel contenu publier ?
- Quelles règles appliquer ?
- Quel rendu graphique ?



Définir le générateur : réponses

- **Format** : OD : présentation papier, plaquette XHTML ; site web :CD-rom, ...
- **Structure** : organisation du contenu dans la page et dans le document;
- **Contenu** : fiche de synthèse, site d'information complet, ...
- **Règles ergonomique** : table des matières, plan, synthèses, voir aussi, liste des définitions, ...
- **Rendu graphique** : Définition de la charte graphique : couleurs, images, ...



Convention de nommage

Souvent, générateurs et conteneurs de styles ont le même "nom":

- paper.generator -> paper.skin.odt
- dossier.generator -> dossier.skin.odt
- web.generator -> web.doss
- epure.generator -> epure.doss

Ce n'est pas une règle.

Les noms peuvent être différents.



Compléter le générateur OD



Code : hwOd



Name : Générateur OpenDocument Text



modelRoot : pointer vers **helloworld.model**

modelRoot : Déclarez ici les modèles autorisés en tant que racine* dans le générateur. Vous n'êtes pas obligé de déclarer tous les **.model** s'ils sont utilisés en tant que sous-modèles.



Créer un transformer list par clic droit sur la référence d'item vide au niveau de **sm:transformeList de nom **hwOd.transflist****

Cet item vous permettra de référencer toute les règles de publication.



Créer un template OD par clic droit sur la référence d'item vide au niveau de **sm:odTemplate de nom **hwOd.odtemplate**.**



Enregistrer



Attention

Il reste une croix rouge au niveau du Skin. Laissez la pour le moment.



Déclaration des règles



Compléter le hwOd.transflist



Créer trois balises <sm:transformer>



Référencer pour les deux premières les deux transformers : */base/sTxt_od.transf* et */base/sTitle_od.transf*



Pour la troisième balise, créer par clic droit un item de type *Publication* -> *Générateurs* 'open document' -> *Transformers* *OpenDocument* -> *compositionOdtTransf*. Avec le nom *helloworld.transf*



Enregistrer





Transformer

Un transformateur (🔧**.transf**) est une *règle de publication* d'un modèle (🔧**.model**) dans un format donné (html, odt, xml etc).

Ils sont utilisés pour traduire une donnée écrite par l'auteur en donnée à l'intérieur d'un document publié. Dans les cas les plus simples, on doit en créer un par **.model** et par générateur. Ils sont référencés par des listes de transformateurs (fichiers **.transflist**) pour pouvoir être exploité par le générateur.

Paramétrage des règles de publications



Les transformateurs de type « Composition » doivent systématiquement être liés à un **.model** de même type.



Se placer dans l'item.



Lier le transformateur à son **.model : helloworld.model**



Dans la balise **sm:content, créer une règle de détection **sm:for codes="code_de_la_part"**.**



Attention

Remplacer **code_de_la_part** par le code de la partie à publier. On le trouve dans le **.model** associé.

Dans notre cas, nous voulons publier la part de code « hello », nous allons donc mettre **sm:for codes="hello"**



Paramétrer l'action à faire : créer un titre structurant (qui peut apparaître dans une table des matières dans le futur). Utiliser **sm:heading**



Au niveau du paramétrage du titre du **heading, on veut afficher, « appeler », le titre de la partie : **sm:partTitle****



Complément

Ce titre est défini par les métas de la part hello de **helloworld.model** et pointe vers **sTitle.model**.

L'« appel » du titre revient à chercher le contenu renvoyé par **sTitle_od.transf** contenu dans le modelelet base, pointé par **hwOd.transflist**.



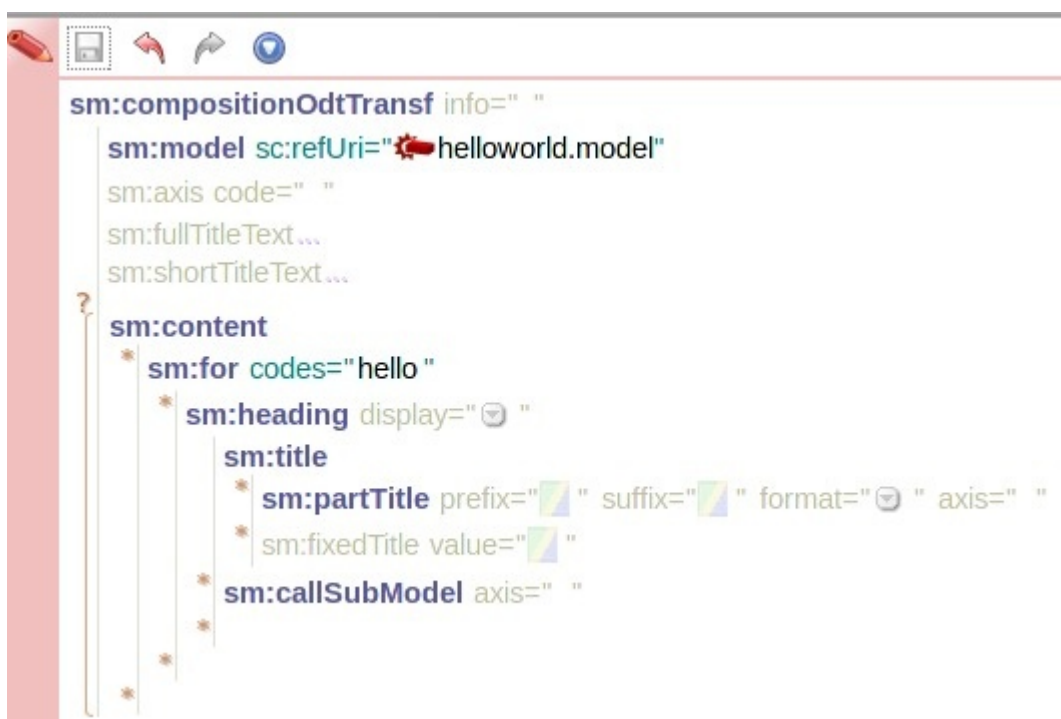
Afficher le contenu de la partie : sous `sm:title`, mais toujours dans `sm:heading`, utiliser `sm:callSubModel`



Complément

Le contenu est modélisé par **sTxt.model** (on le sait car sous la part **hello**, il y a un **allowedModel** qui pointe vers cet item).

L'appel au sous-modèle revient à chercher le contenu renvoyé par le transformer **sTxt_od.transf**.



```
sm:compositionOdtTransf info=" "  
  sm:model sc:refUri="helloworld.model"  
  sm:axis code=" "  
  sm:fullTitleText...  
  sm:shortTitleText...  
  ?  
  sm:content  
    *  
    sm:for codes="hello "  
      *  
      sm:heading display="☺ "  
        sm:title  
          *  
          sm:partTitle prefix=" " suffix=" " format="☺ " axis=" "  
          *  
          sm:fixedTitle value=" "  
          *  
          sm:callSubModel axis=" "
```

Écran 57



Les transformeurs dans les modèles plus compliqués

Pour notre projet helloworld, le réseau de **.transf** correspond exactement au réseau **.model** et à notre unique `<sm:part>` de contenu. Ce n'est pas le cas pour tous les modèles plus complexes, vous pouvez avoir par exemple, des transformeurs qui ne vont pas détecter toutes les parts comme dans le diaporama d'OptimOffice où des commentaires pour l'orateur n'apparaissent pas dans les publications pour le lecteur. Vous pouvez les réorganiser, par exemple les champs d'état civil d'un CV ne seront pas forcément publiés dans le même ordre que celui de l'interface d'édition, en plaçant les `<for>` dans un ordre différent dans le transformeur.

Gabarit, Template



Template

Le template permet de poser la structure des pages : en-têtes et pieds de page par exemple.

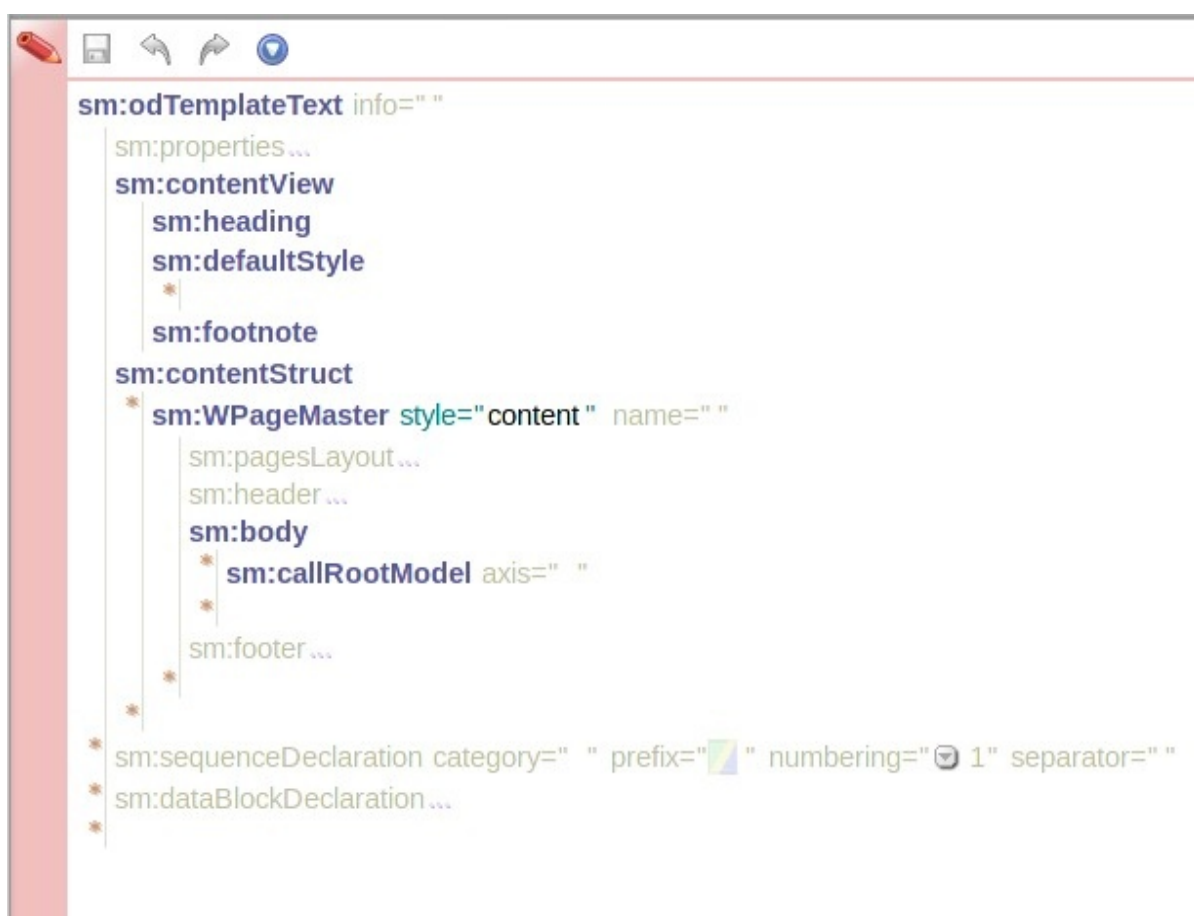
Pour les publications papier, il joue le rôle d'item transverse, alors que pour les publications web c'est l'uiFrame qui joue ce rôle.

Les templates définissent les propriétés du document publié (pour les publications OpenOffice) ou d'un type de page publiée (pour les publications HTML), et posent tous les blocs définis en dehors des transformeurs, à un niveau "au dessus" du contenu.

hwOd.odTemplate

Préoccupez vous maintenant de lier votre racine de contenu à une page. Nous allons faire au plus simple dans ce template de document :

- un seul type de page ;
- **juste un appel au modèle racine du générateur** `<callRootModel>`.



Écran 58

2.3. Stylage



styles

Créer un style, c'est appliquer une charte graphique aux éléments de contenu et du template. Techniquement, le style est constitué d'un fichier **.skin.odt** pour les générateurs OpenOffice ou un ensemble de fichiers **.css** pour les générateurs HTML.



Construire les styles



Se placer dans le générateur OD



Ouvrir la vue **Styles**



Construire



Révéler



Récupérer en glissant déposant le **.skin.odt** dans l'atelier



Lors de la première construction, lier le **.skin.odt** au générateur (se placer dans le générateur et glisser déposer le Skin au niveau de **defaultSkin**).



Quand reconstruire le fichier de style ?

À chaque fois que vous modifiez les transformeurs, en particulier si vous en ajoutez ou si vous déclarez des nouveaux codes et que vous souhaitez les restyler, vous devez revenir effectuer une nouvelle construction du fichier **odStyle**, le retélécharger, et le recopier à l'intérieur de votre atelier SCENARibuilder en écrasant l'ancien. À partir du moment où le générateur est bien lié au style, **reconstruire le style n'écrasera pas toutes les personnalisations que vous avez pu y effectuer.**

La reconstruction sert uniquement à rajouter les parties manquantes par rapport aux évolutions de votre modèle dans l'odStyle pour que vous puissiez ensuite les styler. Il n'est pas nécessaire par exemple de le reconstruire après l'étape de personnalisation qui est décrite ci-dessous.



Personnaliser les styles



Ouvrir le fichier de Skin : *clic droit > Ouvrir dans l'éditeur par défaut du système*



Afficher la fenêtre de styles : *Format > Styles et Formatages*



Dans le menu en bas de la fenêtre de styles, choisir **Tous les styles**



Se placer dans le paragraphe **Style du texte par défaut**



Dans la fenêtre de styles, clic droit sur le style de paragraphe surligné > *Modifier*



Effectuer des modifications telles que : couleur et police de caractère.



Enregistrer



Ne pas sélectionner

Passez toujours par les modifications du style, ne sélectionnez pas directement le texte pour lui attribuer une mise en forme : elle ne serait pas réutilisée. Passez toujours par les styles.

2.4. Tests et réorganisation



Conseil

Recompiler un atelier de test pour tester le nouveau générateur.



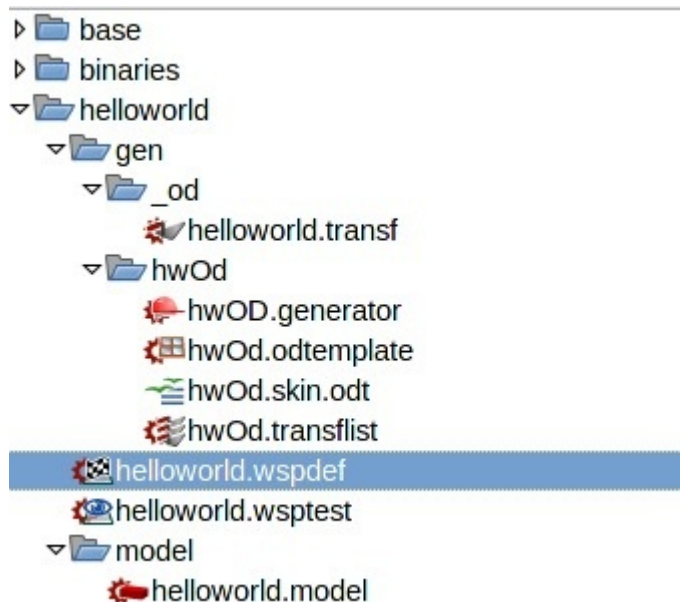
Bonne pratiques

Réorganiser les espaces de l'atelier va vous permettre de mieux progresser dans son développement.

1. un espace model pour placer tous vos .model
2. un espace gen qui contient



- 2.a. un sous-espace `_od` qui contient les transformers
- 2.b. un espace du code du générateur qui contient
le générateur, son template, son translist et son skin



Écran 59

3 Générateur HTML



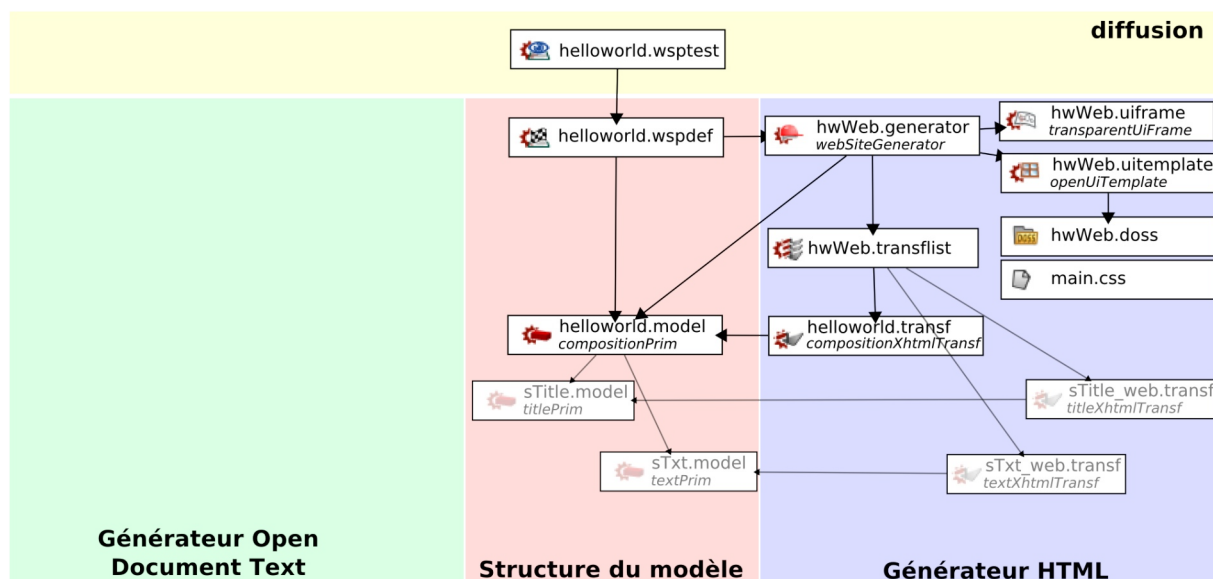
Conseil

Si vous avez réussi la partie précédente "Générateur OD" qui est vivement recommandée en tant que pré-requis à celle-ci, vous n'aurez aucun mal à faire un générateur HTML.

3.1. Mise en place

Schéma de fonctionnement

Les éléments du schéma pour un générateur HTML sont très semblables à notre précédente réalisation pour la publication OD.



Les différences entre générateurs HTML et OD

Pour effectuer une comparaison (appliquée à notre exemple helloworld) :

- Le type et le contenu de tous les items sont spécifique à la publication HTML.
- L'item de styles est remplacé par un dossier (ou plusieurs) contenant **les css et les images de stylage**
- Un item **uiservice** fait son apparition.



Attention

Vous ne pouvez pas copier coller, mélanger des items de publication XHTML avec des items de publication OpenDocument.

Création du générateur



Truc & astuce

Nous allons créer le générateur XHTML HwWeb à partir du **wspdef**

> "cf Mise en place", page 22.



Se placer dans le wspDef.



Créer une nouvelle balise `sm:genBox` sous la balise existante



Par clic droit sur l'élément vide au niveau de `sm:generator`, *Créer un nouvel item*.



Choisir *Publication* > *Générateurs 'web'* > *webSiteGenerator*



Le nommer `hwWeb` et Créer



Enregistrer

```
sm:wspDefinition keyWsp="helloworld" name="Salut le monde" defaultWspCode=""
majorVersion="0" mediumVersion="0" minorVersion="1"

sm:publicClasses
  sm:class sc:refUri="helloworld.model (/helloworld/model)" access="readWrite" sortKey=""
    sm:mainView
      sm:formEditorTab name="Edition" idInView="" display="visible" formEditorKey=""
      sm:blocksTab name="Publication" display="visible" idInView=""
      sm:generatorsBlock name="Publication" idInView="" displayMode="lockedOpen"
        sm:genBox name="" idInView=""
          sm:generator sc:refUri="hwOd.generator (/helloworld/gen/od)"
          sm:deployment...
        sm:genBox name="" idInView=""
          sm:generator sc:refUri="hwWeb.generator (/helloworld/gen/web)"
          sm:deployment...
      sm:defaultMainView ...
    sm:generatorSkins ...
  sm:wspUi ...
  sm:wspConfig ...
  sm:overlays ...
  sm:migration ...
```

Écran 62



3.2. Composants et paramétrage du générateur

Le générateur



Paramétrer le générateur



Renseigner le code et le name avec `hwWeb` et « Générateur Xhtml »



Lier au `.model` racine



Choisir le standard



Créer un `transparentUiFrame` `hwWeb.uiframe`



Attention

Ne changez pas le contenu du fichier créé, il est utilisé pour les services spéciaux (scorm, assessements...) dont nous n'avons pas besoin pour cet exemple.



Créer un `transformerList` `hwWeb.transflist`



Créer un template de code `main` et de type `openUiTemplate` `hwWeb.uiemplate`



Dans un `webSiteGenerator`, le `template` correspond à un `layout` de page (l'agencement des éléments HTML à l'intérieur de la page). Chaque `template` est associé à un `code`.



```
sm:webSiteGenerator code="hwWeb " name="Générateur XHTML "  
  sm:icon sc:refUri=" " "  
  sm:modelRoot  
    * sm:allowedModel sc:refUri="helloworld.model (/helloworld/model)"  
    *  
  sm:mode standard="HTML 4.01 Transitional (Standards mode)" debug="no"  
  sm:uiFrame sc:refUri="hwWeb.uiFrame"  
  sm:transformerList info=" " sc:refUri="hwWeb.transfList"  
  *  
  sm:uiTemplateList  
    * sm:uiTemplate code="main " sc:refUri="hwWeb.uiTemplate"  
    *  
  sm:defaultSkin sc:refUri=" " name=" "  
  sm:publishFiles sc:refUri=" "
```

Écran 63

Déclaration des règles

Compléter le `hwWeb.transfList`

Créer trois balises `<sm:transformer>`

Référencer pour les deux premières les deux transformers : `/base/sTxt_web.transf` et `/base/sTitle_web.transf`



Pour la troisième balise, créer par clic droit un item de type *Publication* -> *Générateurs 'web'* -> *Transformers Web* -> *compositionXhtmlTransf*. Avec le nom *helloworld.transf*



Enregistrer

Règles de publications



Paramétrer la version Web de helloworld.transf

La structure souple d'un site web nécessite de déclarer **chaque nouvelle page** et d'associer cette page à un **template**, en reprenant le même code que celui spécifié dans le *webSiteGenerator*.



Recopier le paramétrage

```
sm:compositionXhtmlTransf info=" "  
  sm:model sc:refUri="helloworld.model (/helloworld/model)"  
  sm:axis code=" "  
  sm:fullTitleText ...  
  sm:shortTitleText ...  
  ?  
  sm:navigation  
    * sm:page template=" main " step=" "  
      sm:title  
        * sm:fixedTitle value="Hello World ! "  
      sm:outlineClasses ...  
      * sm:mainZone format="xhtml"  
        * sm:for codes="hello "  
          * sm:WHeadingBlock widgetClass=" " crossRefEntries=" " class="box "  
            sm:title  
              * sm:partTitle prefix=" " suffix=" " format=" " axis=" "  
              * sm:fixedTitle value=" "  
              * sm:callSubModel axis=" "
```

Écran 64



Choisir le code du template, c'est celui qui a été défini dans le générateur.



Bien définir le code de la part à afficher, tel qu'il a été défini dans le .model.



Truc & astuce

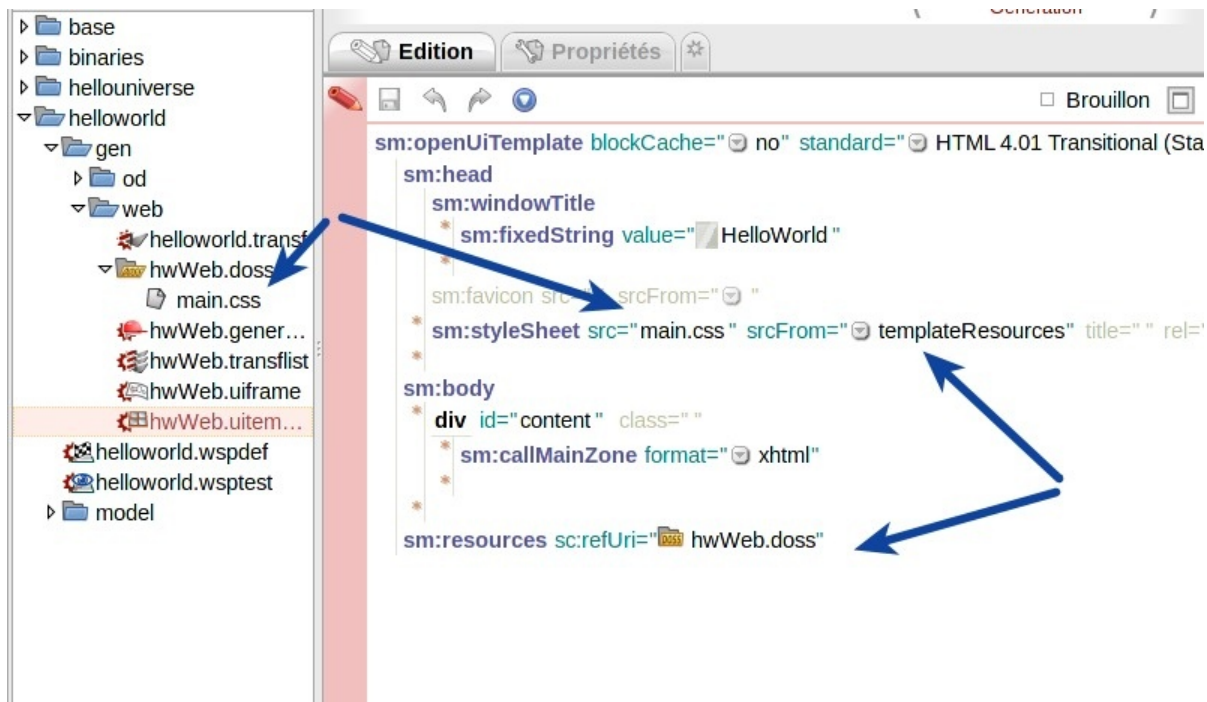
Il est aussi possible d'utiliser `for codes="*"` qui permet de sélectionner toutes les parties existantes dans le .model.



Contenu du for

En contenu du `<for>`, on déclare un `<WHeadingBlock>`. Lorsque l'on spécifie un `class` dans un bloc, cela va rajouter dans l'HTML produit un attribut `"class=..."` que l'on pourrait exploiter à l'étape du stylage.

Paramétrer le template



Écran 65



En écrivant un `<openUiTemplate>`, vous décrivez assez librement la structure de votre page web :

- ▶ En entête, spécifiez un nom de fichier du type **main.css** qui sera extrait du répertoire de ressources **hwWeb.doss** que vous devez créer aussi.
- ▶ A l'intérieur du body, cliquez sur l'étoile d'insertion et sélectionnez *éléments* pour faire les `<div>` suivant vos désirs de mise en page. Ajoutez un `<callMainZone>` pour que le réseau de transformeurs prenne la main sur la publication du contenu.
- ▶ Enfin allez au niveau de la balise `<resources>`, faites un clic droit sur l'item vide se situant à côté de cette balise puis cliquez sur *Créer un nouvel item* et choisissez *Ressources -> Dossier de ressources (.doss)*



srcFrom

Pour chaque css à utiliser, vous devez spécifier au template d'où vient cette CSS. Dans le cas d'un dossier de ressources lié au template, on choisit **templateResources**

> cf *pageErreur* : source de la référence non trouvée.

3.3. Stylage

main.css

A l'aide de votre éditeur de texte favori créer le fichier **main.css**. Ensuite, glissez-déposez le fichier **main.css** dans l'atelier à l'intérieur du répertoire **.doss** spécifié auparavant dans le template. Cette partie fait appel à vos connaissances HTML et CSS, mais voici un petit exemple :



```
main.css X
1 body{
2     margin:20px;
3 }
4 h1{
5     background-color:#DDDDFF;
6     padding:0.3em;
7     border-bottom:3px solid #777799;
8     margin-bottom:0.5em;
9     margin-top:0.5em;
10 }
```

Écran 69



Tester plus vite les résultats des CSS

Vous pouvez faire une génération HTML dans **SCENARitest**, et modifier le fichier CSS dans cette génération pour voir les modifications en "temps réel", puis une fois satisfait du résultat, recopiez ce fichier CSS dans votre répertoire **.doss** de builder. Soyez juste prudent car **les générations sont écrasées à chaque nouvelle régénération**.



Retrouvez plus vite les noms des classes dans le document

Si vous utilisez firefox, vous avez la chance de pouvoir profiter d'extensions bénéfiques à tout développeur web.

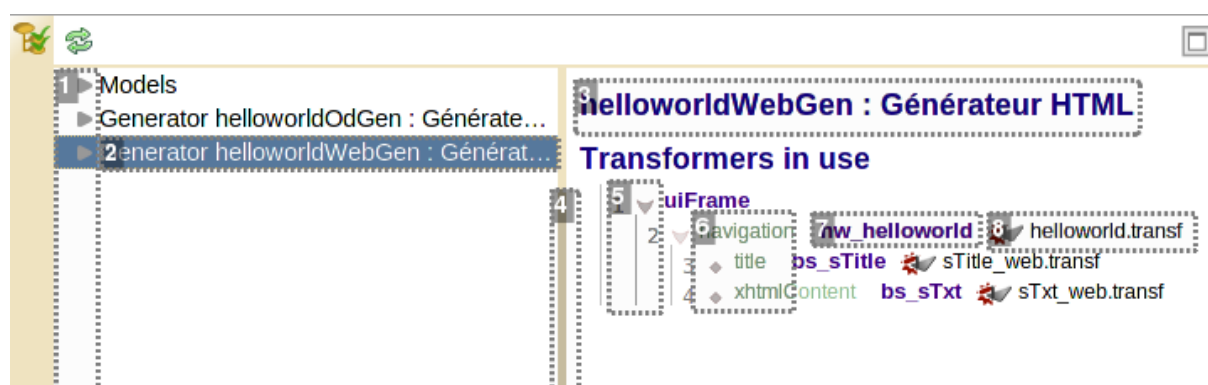
Firebug : ouvrez le par l'icône dans la barre d'état, choisissez l'onglet HTML, inspect, et cliquez sur un élément de la page.

3.4. Tests et réorganisation



Conseil

Recompiler un atelier de test pour tester le nouveau générateur.



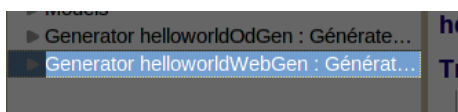
1-



Liste des éléments à contrôler

Dans cette zone sont listés tous les éléments à tester : la modélisation ainsi que les générateurs.

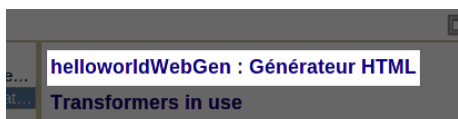
2-



Sélectionner un élément à contrôler

Il suffit de cliquer sur le nom de l'élément à contrôler pour voir son analyse.

3-



Code et Name du générateur contrôlé

Permet de vérifier le générateur que l'on contrôle

4-

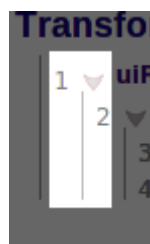


Barre de séparation modulable

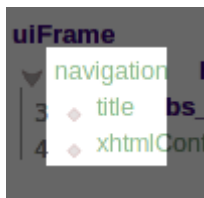
Il est possible de déplacer cette barre pour mieux visualiser la zone de contrôle.



5-



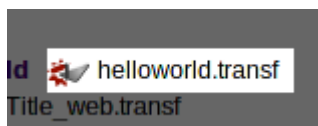
6-



7-



8-



Liste des transformers attendus

Ici sont listés, en partant du départ, l'uiFrame (qui fait un callRootModel pour créer l'index.html) et tous les transformers qui sont attendus.

Catégorie du transformer

- navigation : pose une page physique html.
- title : renvoie une chaîne de caractères brute.
- xhtmlContent : contenu html pure (hors meta/head/body)

Identifiant du model

Préfixe et code du modèle qui est interrogé.

Transformer trouvé

Un transformer a été trouvé (préalablement déclaré dans le générateur/dans le transflist).



Truc & astuce

Afficher sous forme de lien cliquable, n'hésitez pas à cliquer dessus pour parcourir vos transformers.



Bonne pratiques

Réorganiser les espaces de l'atelier va vous permettre de mieux progresser dans son développement.

1. un espace model pour placer tous vos .model
2. un espace gen qui contient
 - 2.a. un sous-espace _web qui contient les transformers
 - 2.b. un espace du code du générateur qui contient le générateur, son template, son transflist, son uiFrame et le .doss de styles



4 Diffusion

Redistribuez des wsppacks

Votre helloworld n'aura peut-être pas des millions d'utilisateurs partout à travers le monde, mais dans l'optique de réaliser des futurs modèles plus importants, vous allez apprendre à les redistribuer aux **auteurs**. La première option est de fournir un **wsppack**, que l'utilisateur va pouvoir installer sur **SCENARichain**.

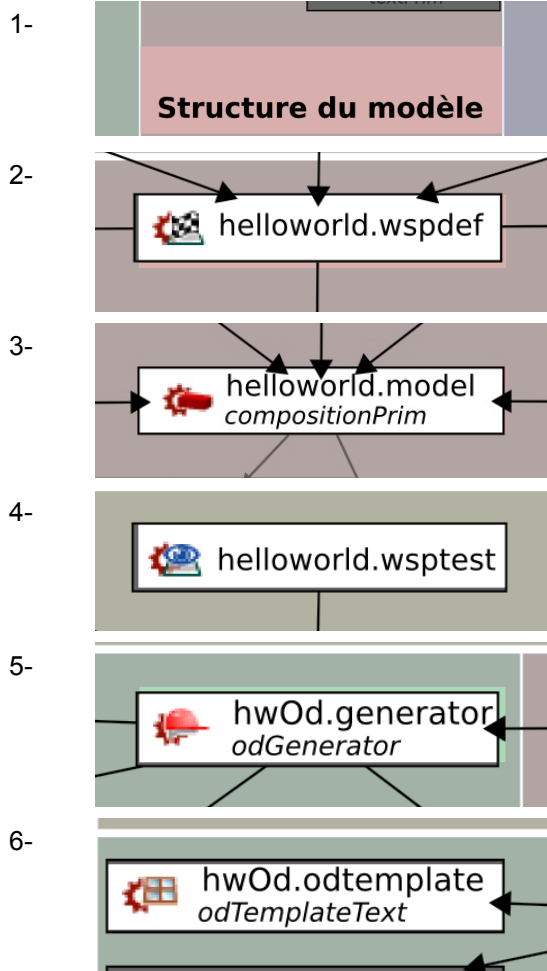
Rien de plus simple : créez un nouvel item **helloworld.packMake**, et comme vous l'avez fait pour le **wsptest**, un glisser-déposer du **wspdef** vous permet d'activer la vue Compilation puis de lancer l'export de votre atelier et d'obtenir le fichier **helloworld_0.0.1.wsppack**.

Redistribuez des applications

SCENARI permet aussi de transformer une ou plusieurs chaînes éditoriales en une application standalone*. Vous devez pour cela créer un fichier **appmake** et utiliser un **sckit**. Vous obtenez un programme pour linux, windows et/ou mac. Ceci sort du cadre de notre simple exemple helloworld.

5 Bilan

Mission accomplie !



Analyse

> "cf HelloWorld", page 7.

> "cf Structure du modèle", page 9.

Définition de l'atelier

> "cf Mise en place", page 9.

```
. model
```

> "cf Composants du modèle", page 12.

SCENARitest

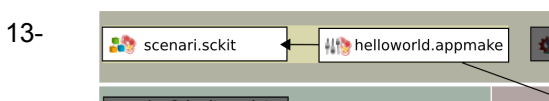
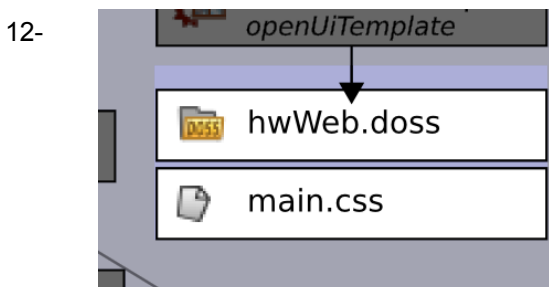
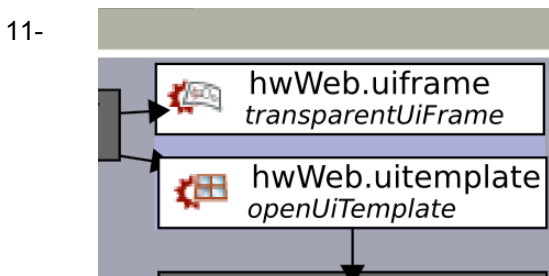
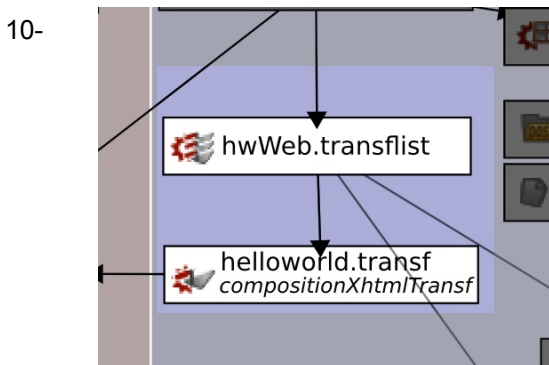
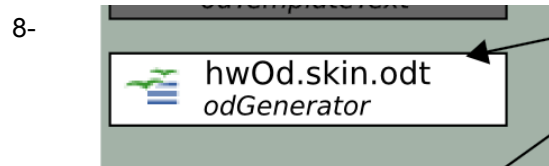
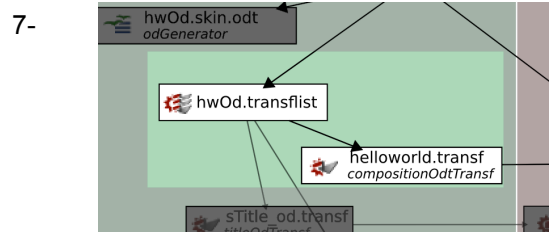
> "cf Tests de votre chaîne", page 19.

Générateur OD

> "cf Mise en place", page 22.

Template OD

> "cf Composants et paramétrages du générateur", page 25.



Transformers OD

> "cf Composants et paramétrages du générateur", page 25.

Styles OD

> "cf Stylage", page 32.

Générateur Web

> "cf Mise en place", page 35.

Transformers Web

> "cf Composants et paramétrage du générateur", page 38.

Templates Web

> "cf Composants et paramétrage du générateur", page 38.

CSS

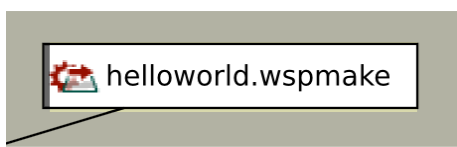
> "cf Stylage", page 42.

AppMake

> "cf Diffusion", page 46.



14-

**PackMake**

> "cf Diffusion", page 46.

Voulez vous continuer ?

La suite du programme si vous souhaitez réaliser des modèles sérieux est la maîtrise de nouveaux concepts ou items, vous avez quelques exemples pour vous aider dans les Modèles disponibles sur scenari-platform.org.

Savoir analyser la structure d'un document

cette présentation helloworld est très technique et ne vous apprend pas à prendre du recul pour créer un réseau de .model.

Les dataform

elles se substituent aux primitives compositionPrim si vous souhaitez traiter des champs de données (par exemple les information d'état civil dans CV)

Les autres primitives binaires

pour inclure des images, du son, et paramétrer leur transformation.

La DTD complète des transformeurs

dans cet exemple nous n'avons utilisés les transformeurs qu'à 10% de leur capacités.

Les axis

à l'intérieur du même générateur, il est possible de créer plusieurs version d'un même transformeur.

Particularités du support Open Document

table des matières, indexes, annexes, référence des illustrations, entêtes et pieds de pages, recto-verso...

Particularités du support HTML

les widget de menu, le fonctionnement des zones.

HtmlToOd

mettre un pdf à partir d'une page web.

Les genDeploy

upload *FTP* des générations ou autres actions de déploiement.



Le stylage de l'éditeur

faire des petites modifications sur le style des parties de contenu dans l'éditeur et les *icônes* des items.

Exploiter les extension

sound, assessements, SCORM

III HelloUniverse

Introduction

Dans cette seconde partie, nous allons utiliser notre expérience HelloWorld pour aller plus loin, et créer un "HelloUniverse". Nous nous plaçons dans une **problématique de modélisation** : créer une mini-référence des planètes de l'univers. Puis, ce sera également l'occasion de faire des **publications plus originales**, un vrai "mini site", une vraie "mini publication" papier.

Après helloworld, modèle qui ne fait aucun sens et dont aucun auteur n'aurait besoin, nous adoptons le thème : faire un modèle pour décrire des planètes de l'univers. *The world is not enough !*



Objectifs

Dans ce tutoriel, nous allons apprendre à :

- Utiliser quelques pistes pour **se représenter** un modèle complexe ;
- Traiter les données de type **formulaire** avec les *DataForms*, les publier sous forme de tableau avec des images ;
- Concevoir un site HTML **multi-pages** avec un menu, et un document ODT avec une **table des matières** ;
- Décliner un transformer en différents **axis** en fonction du *contexte* ;
- Trouver des informations sur le **stylage de l'éditeur** et la diffusion sous forme d'**application autonome**.



1 Méthodologie

1.1. Analyse préliminaire

Lorsqu'un concepteur de base de données se lance dans la création d'un schéma de données, il va réfléchir sur les informations et leur place logique avant de faire une mise en contexte technique. De même, en ingénierie documentaire, il est important de se représenter **la structure globale du modèle** avant de passer à la conception dans SCENARBuilder.



Remarque

La conception de modèles de documents est transverse à plusieurs métiers et contextes d'usages. Elle demande à la fois une bonne connaissance de la modélisation documentaire mais avant tout une connaissance du contexte d'utilisation de ce futur modèle. Plusieurs méthodes d'approches peuvent être envisagées, nous vous proposons ici un exemple d'une approche qui n'est à considérer ni comme exhaustive ni comme représentative des méthodes de conceptions.

Première étude

Étude du domaine

Domaine exemple utilisé pour ce tutoriel : les planètes.

Nous voulons décrire l'univers et ses planètes.

Documentez vous sur le thème de votre modèle. Vous devez devenir un expert des planètes et de tout ce qui se trouve dans l'univers pour pouvoir être à l'aise avec les choix de modélisation de ces données. Soyez attentif aux informations de ces documents à modéliser : y a-t-il souvent des parties de textes qui ont le même type de contenu pour toutes les planètes ? Devant une masse très volumineuse d'informations, lesquelles doivent être présentes sur vos documents et lesquelles ne vous intéressent pas ?

Partir de documents existants est une source d'inspiration très précieuse, ci-dessous des liens vers des documents pour votre modèle "helloUniverse":

- <http://fr.wikipedia.org/wiki/Plan%C3%A8te>
- <http://fr.wikipedia.org/wiki/Astre>
- <http://fr.wikipedia.org/wiki/Univers>
- [http://fr.wikipedia.org/wiki/Mars_\(plan%C3%A8te\)](http://fr.wikipedia.org/wiki/Mars_(plan%C3%A8te))
- http://fr.wikipedia.org/wiki/Mercure_%28plan%C3%A8te%29#_note-3



Identifiez les données

À partir de votre étude, identifiez les "objets" à utiliser, la structure générale de vos données. Soyez raisonnable et n'hésitez pas à **simplifier** quand vous le pouvez.



Truc & astuce

Posez vous des questions :



Dans notre cas, nous avons différents types de corps célestes mais avons-nous besoin de les distinguer par 2 types d'objets différents ?



Lors de la rédaction, l'auteur a-t'il besoin d'avoir des types de parties très différentes dans le cas d'une étoile ou dans le cas d'une planète ?



Doit-on créer un type d'item spécifique pour les "planètes naines" ?

Nous allons choisir la solution la plus simple : le même type d'item est utilisé par l'auteur pour les étoiles, les planètes et les planètes naines : la structure sera la même mais à l'intérieur du contenu, rien ne nous empêche de prévoir un moyen pour l'auteur de spécifier le type de l'astre. Il n'y a pas de bonne réponse ou de mauvaise, c'est vraiment en fonction des besoins.



Autre question à se poser : doit-on créer des objets comme les galaxies ou les systèmes ?

Raisonnablement, nos auteurs n'ont pas trop de connaissances à capitaliser pour des planètes en dehors du système solaire, donc nous allons nous limiter à celui-ci dans un premier temps et nous n'avons pas besoin de représenter ces éléments. Si nous avons à placer d'autres étoiles, il serait intéressant de séparer les astres du système solaire des autres, donc d'adopter une structure en galaxie / systèmes.



Souhaitons nous regrouper les informations des planètes ?

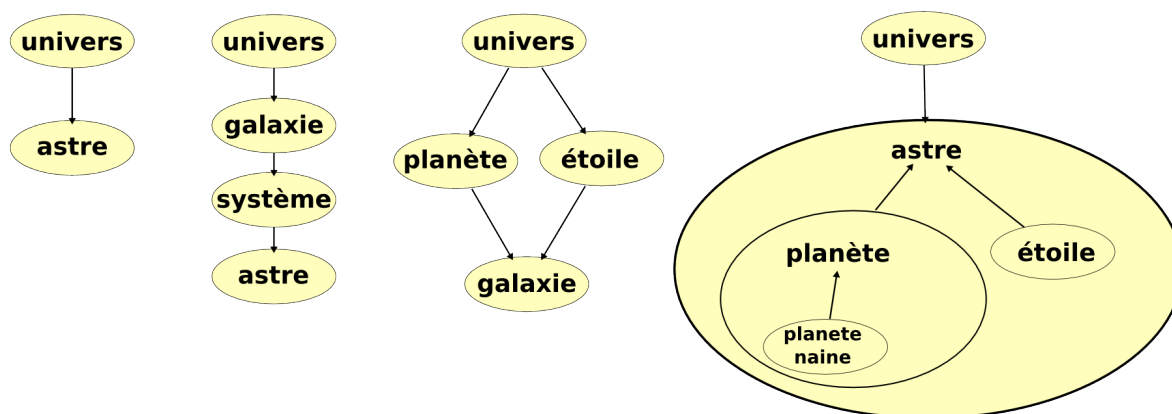
Nous décrivons plusieurs planètes et **il leur faut impérativement un conteneur**, nous allons les placer dans une racine **univers**.



Exemple

Quelques exemples de structures possibles, nous choisissons la première qui offrira largement assez de complexité à cet exemple.

Différentes structures possibles suivant les besoins



Lors de la conception technique, vous pourrez être amené à découper ces objets en plusieurs **.model** SCENARBuilder mais vous avez déjà préparé cela par l'étape abstraite du travail.

Contexte du projet



Attention

Il est important de se poser des questions sur le contexte du projet, son but, les futurs auteurs et les futurs lecteurs.

Formaliser ces points vous permettra d'y voir plus clair dans le projet et de vous aider dans vos choix de modélisation ou de publications.



Approfondissement de l'étude



Définir le contenu des objets

L'univers contient des planètes, nous ne lui associerons pas d'autres données.

Chaque astre est associé à une série d'informations précises :

- nom
- photo
- nature
- la liste de ses satellites et leurs distances orbitales
- diamètre
- masse
- période de rotation
- période sidérale



Ajouter du contenu textuel riche : sous forme de paragraphe pour décrire les planètes. Question à se poser sur les paragraphes : peut-on les pré-définir (forcer un titre ? un ordre ?) ou est-ce à l'auteur de les écrire librement.

Les choix suivants s'offrent à nous

1. **Liberté total de l'auteur sur les paragraphes** : on laisse choisir à l'auteur le rôle de chaque paragraphe et leur nombre sans lui proposer des parties spécifiques.
2. **Sans ordre précis, parties libres et éventuellement parties prédéfinies** : l'auteur crée des parties comme bon lui semble, mais il peut utiliser certaines parties avec un rôle prédéfini.
3. **Parties prédéfinies en ordre fixe + parties libres** : on spécifie d'avance les rôles de 4 paragraphes et l'auteur peut en ajouter librement à la fin.
4. **Restriction total des parties** : nous créons et choisissons d'avance les rôles de 4 paragraphes (avec un titre par défaut ou même un titre fixe).



Identifier des structures récurrentes :

Essayant d'identifier les besoins de nos auteurs, nous allons identifier **4 parties qui reviennent couramment** : introduction, atmosphère, surface, noyau.

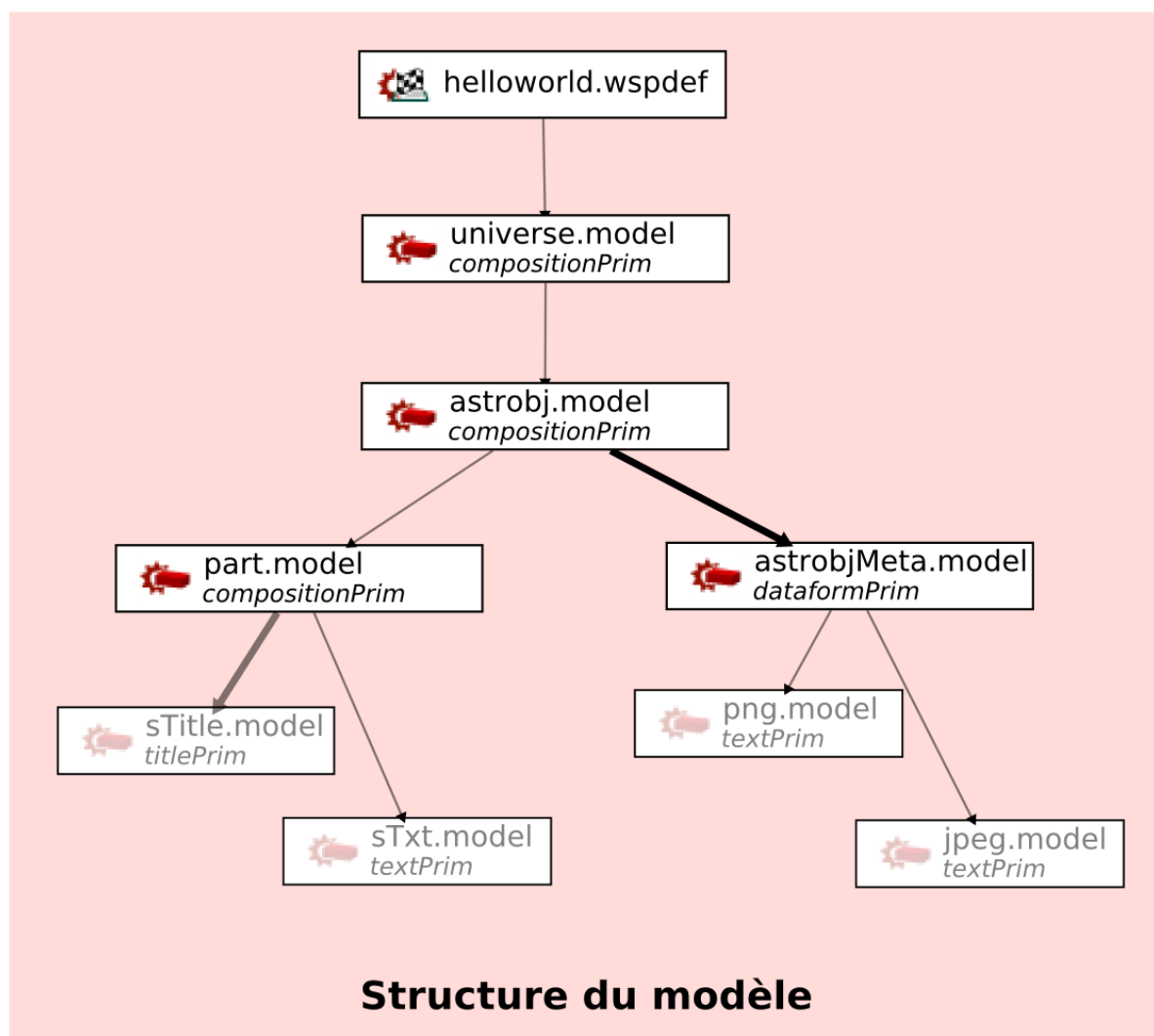


Attention

Mais chaque planète a des particularités que l'auteur peut souhaiter décrire dans des paragraphes qu'il crée lui même et dont on ne peut pas forcément anticiper le rôle. Nous retenons donc la solution « **Parties prédéfinies en ordre fixe + parties libres** ».

Schématisation

Nous allons au fil de ce tutoriel construire un réseau de fichiers **.model** respectant ce schéma (en plus des autres items ici masqués par soucis de simplification). Nous avons vu que l'on pouvait décomposer notre domaine en objets, mais aussi **identifier des types de parties** répétés à l'intérieur d'un modèle ou les re-décomposer, ce qui explique ce niveau supplémentaire en dessous de **astreobj.model**.



1.2. Conseils



Ne pas commencer par trop compliqué

Lorsque vous faites un document, déterminez les **concepts qui peuvent être mis en commun** : par exemple, pour nos 4 parties "introduction, atmosphère..." , leur structure est identique. Il ne faut donc pas faire 4 **.model** différents pour représenter chacun, mais un **.model** global qui va couvrir les 4 cas d'usages.



Approfondissement des notions



Conseil

Il vous faudra découvrir quelques autres notions de modélisation mais cette analyse peut se poursuivre. L'objectif étant d'anticiper le plus possible les difficultés, si vous maîtrisez bien SCENARiBuilder et que vous savez ce que vous voulez, vous pouvez vous en sortir en y continuant les essais.



Conception pour un tiers

Mais si vous concevez un modèle en fonction des besoins d'un tiers sur un sujet que vous connaissez mal, il vous faudra être bien plus spécifique dans les questions que vous allez lui poser, et obtenir des indications comme les types de données, le format des ressources, leur hiérarchie, etc.

Il y a des choix aisément modifiables par la suite, comme autoriser ou interdire l'internalisation, modifier un nom de champ. Avec l'expérience vous pourrez identifier ce qui est crucial pour un projet de ce qui est de l'ordre du détail. Même si c'est plus simple et plus gratifiant de concevoir parfaitement du premier coup, il est plus sûr de réaliser son modèle par étape et de le confronter à d'autres utilisateurs ou au tiers en question plus haut **de valider et de réagir** sur votre modélisation.

Méthodes de schématisation



Remarque

Il n'y a pas de méthode de représentation de modèle documentaire qui ait satisfait unanimement l'ensemble des modélisateurs experts Scenari. En effet, aucune schématisation unique n'est aussi riche que la conception SCENARiBuilder. Utilisez cette schématisation pour initialiser votre projet, communiquer avec des personnes extérieures au monde Scenari, documenter votre projet.

Vous choisirez une méthode ou une autre en fonction :

- de vos exigences professionnelles.
- de l'aboutissement de la conception : un modèle évolue beaucoup en phase de conception, il n'est pas forcément nécessaire de décrire chaque champ chaque possibilité dès le début de la conception. Cela peut entraîner une perte de temps et un inversement des priorités (amélioration de la modélisation VS maintenance de la documentation technique).
- de vos habitudes de travail.
- des personnes avec lesquelles vous aller échanger autour de ces schémas. Si vous travaillez seul,



c'est tout de suite moins contraignant.

Méthodes reconnues

La modélisation **UML**, choisie pour ce tutoriel (voir plus loin).

La **Carte Heuristique** – *mind map*. Cette méthode est intéressante en phase d'analyse pour discuter avec le tiers non spécialiste en ingénierie documentaire.

Un arbre, complété d'esquisse représentant l'application du point de vue de l'auteur.

La modélisation directement dans l'IDE SCENARiBuilder. Rapide et immédiate, essentiellement quand l'analyse initiale est triviale et que les premières questions de modélisation sont très précises. Cette modélisation permet d'avancer rapidement dans le projet mais peut vous faire passer à côté de points essentiels.



Le choix pour ce tutoriel : UML

Les schémas fournis dans ce guide ont un rôle de démonstration, il est aussi possible de faire des diagrammes "type UML" Scenari pour représenter le modèle avec plus d'informations. Il n'existe pas (encore) d'outil pour les générer automatiquement à partir d'un modèle SCENARiBuilder – et encore moins de générer un modèle SCENARiBuilder à partir d'un schéma. Notez qu'il est tout de même possible d'exporter la DTD du modèle sous forme de **schéma RelaxNG** (*créer un item* > *Gestion des ateliers* > *Compilation* > *wspRelaxNg*).

2 Mise en place

2.1. Structure



Introduction

Dans cette partie, nous allons copier HelloWorld sous le nom HelloUniverse, et travailler sur cette copie en ajoutant et modifiant des fichiers **.model** pour que l'éditeur Scenari prenne en compte nos nouveaux paragraphes de données.



Initialisation

Point de départ



Vous partez du même atelier que celui du tutoriel précédent

Copiez le répertoire **helloworld** à la racine de votre atelier (cliquez sur son titre pour sélectionner la racine). Collez ce répertoire à la racine de votre atelier et renommez-le en **helloUniverse**.



Ensuite, quelques renommages sont à effectuer

1. **helloworld.packmake**, `.wspdef`, et `.wsptest` -> **hellouniverse.packmake** etc.
2. Changer la clé `keyWsp` et le `name` du `wspdef`.
3. Renommer **helloworld.model** en **universe.model** et changer son `code` et son `name`
4. Renommer les deux transformers **helloworld.transf** (od et web) en **universe.transf**
5. Renommer les dossiers des générateurs, les générateurs, leur skin, template, transflist
6. Changer le code des générateurs.



Vérification

Vérifiez **l'écran de contrôle** et assurez vous par test que votre copie est pleinement fonctionnelle, en compilant un SCENARitest



Namespace

Vous n'avez pas besoin de changer les namespaces et prefix* tant que les codes sont uniques. Vous pourriez aussi faire l'inverse : changer uniquement les namespaces mais garder le même code. C'est le couple **namespace + code** qui *identifie un item*.



Conseil

En général on n'utilise qu'un seul namespace par atelier, il identifie le nom du projet dans son ensemble.



Mais pour faire une déclinaison de modèle

Pour faire une déclinaison d'un modèle, vous n'êtes pas obligé de faire toute cette série de renommage, mais vous pouvez aussi copier vos fichiers dans un atelier différent.

Ou prendre le risque de conserver les 2 répertoires côte à côte sans changer les codes, à partir du moment où il n'y a aucune interaction d'items entre les 2 répertoires il n'y aura pas d'erreur. Mais alors, surveillez bien l'écran de contrôle et assurez vous que les auteurs n'utiliseront pas l'autre version du modèle sans avoir conscience des risques ou différences (ceci est toutefois **déconseillé**).



Les librairies

En cas de modèles différents qui partagent des **.model** ou ressources communes, le mieux est de créer un répertoire "librairie" qui contiendra l'intersection des 2 modèles, un ensemble d'items SCENARiBuilder réutilisables à l'intérieur de **.model** d'un même atelier. Ce n'est pas le cas entre nos 2 modèles puisque notre micro-helloworld ne contient qu'un seul fichier **.model**.

Nous utilisons **base** et **binaries** comme des librairies.

Nouvelle composition Prim pour astrobj.model et partie.model



Copiez 2 fois le fichier **universe.model** et nommez les copies **part.model** et **astrobj.model** dans le même répertoire.



Nous allons simplifier l'ensemble "titre + contenu" en une seule composition partie.



Ouvrez le fichier **part.model** et mettez **sTitle** en tant que <meta> générale



Supprimez le set et créez un sm:part à la place, de **code** part, de **name** Partie, toujours internalisé, obligatoire et de modèle **sTxt.model**.

```
sm:compositionPrim name="Partie" category="" info=""
sm:help ...
sm:identification targetNamespace="scenari-platform.org:helloworld" targetPrefix="hw" code="part" itemExtension="xml"
sm:structure
  sm:meta sc:refUri="sTitle.model (/base/sTitle)" usage="optional"
  * sm:part code="part" name="partie" family="" internalized="always" usage="required"
  * sm:allowedModel sc:refUri="sTxt.model (/base/sTxt)" restrictUserDep=""
```

Écran 102



Nous définissons les blocs de texte suivant la méthode que nous avons choisie précédemment : un astre est souvent décrit par 4 parties de texte prédéfinies, et plusieurs paragraphes libres.

```
sm:compositionPrim name="Astre" category="" info=""
sm:help quickHelp=""
sm:identification targetNamespace="scenari-platform.org:helloworld" targetPrefix="hw" code="astrobj" itemExtension="xml"
sm:structure
  sm:meta sc:refUri="sTitle.model (/base/sTitle)" usage="required"
  * sm:part code="intro" name="Introduction" family="" internalized="always" usage="optional"
  * sm:allowedModel sc:refUri="part.model" restrictUserDep=""
  * sm:part code="atm" name="Atmosphère" family="" internalized="always" usage="optional"
  * sm:allowedModel sc:refUri="part.model" restrictUserDep=""
  * sm:part code="surf" name="Surface" family="" internalized="always" usage="optional"
  * sm:allowedModel sc:refUri="part.model" restrictUserDep=""
  * sm:part code="core" name="Noyau" family="" internalized="always" usage="optional"
  * sm:allowedModel sc:refUri="part.model" restrictUserDep=""
  * sm:set usage="optional"
  * sm:part code="part" name="Autre partie" family="" internalized="always"
  * sm:allowedModel sc:refUri="part.model" restrictUserDep=""
```

Écran 103



Les 4 premiers paragraphes sont à l'extérieur du <set>, ce qui force l'auteur à respecter leur ordre. Ils sont en `usage="optional"`, l'auteur peut choisir d'en laisser certaines vides par manque d'information. Contrairement au helloworld, notre titre est placé dans la composition du "sous-modèle" et non à l'intérieur du paragraphe, il est intégré à l'item `part.model`.

Référencer astrobj.model depuis universe.model



Il ne reste plus qu'à modifier notre élément racine pour faire appel à `astrobj.model` (car tout simplement : l'univers contient des astres).

```
sm:compositionPrim name="Univers" category="" info=""
sm:help ...
sm:identification targetNamespace="scenari-platform.org:helloworld" targetPrefix="hw" code="universe" itemExtension="xml"
sm:structure
  sm:meta sc:refUri="" usage="required"
  sm:set usage="required"
    sm:part code="astrobj" name="Astre" family="" internalized="never"
    sm:allowedModel sc:refUri="astrobj.model" restrictUserDep=""
```

Écran 104



Un choix intéressant est d'obliger l'auteur à créer un fichier distinct par planète : `Internalized = never`.

Objet externe

Pour que `astrobj.model` puisse être utilisé en tant qu'objet externe*, il faut qu'il soit dans les <publicClasses> de notre `hellouniverse.wspdef`.



placez vous dans le wspdef



cliquez sur l'étoile d'insertion se situant sous la balise <publicClasses> et choisissez la balise <class>



glissez-déposez le fichier **astrobj.model** sur l'item vide se situant à côté de cette balise.



Sous la balise <publicClasses>, vous avez maintenant deux balises <class> pointant vers les fichiers **astrobj.model** et **universe.model**.

```
sm:wspDefinition keyWsp="hellouniverse" name="Salut l'univers" defaultWspCode=""
majorVersion="0" mediumVersion="0" minorVersion="0"
sm:publicClasses
  sm:class sc:refUri="universe.model (/helloUniverse/model)" access="readWrite" sortKey=""
  sm:mainView
    sm:formEditorTab name="Édition" idInView="" display="visible" formEditorKey=""
    sm:blocksTab name="Publications" display="visible" idInView=""
    sm:generatorsBlock name="Publications" idInView="" displayMode="memorized"
      sm:genBox name="" idInView=""
        sm:generator sc:refUri="huOD.generator (/helloUniverse/gen/huOd)"
        sm:deployment...
      sm:genBox name="" idInView=""
        sm:generator sc:refUri="huWeb.generator (/helloUniverse/gen/huWeb)"
        sm:deployment...
  sm:class sc:refUri="astrobj.model (/helloUniverse/model)" access="readWrite" sortKey=""
```

Écran 105



Tests de la nouvelle modélisation



Conseil

Vous pouvez à nouveau tester votre modèle mais attention : les transformeurs sont "cassés"... Ils ne correspondent plus aux **.model** et les générateurs ne fonctionnent plus. Pour le moment, **seul l'éditeur est fonctionnel**.

2.2. Publication Web



Introduction

Nous avons modifié la structure du modèle, mais les transformeurs utilisent toujours les anciennes données : résultat, le générateur ne fonctionne plus du tout ou ne publie presque rien. Il faut donc aussi modifier les transformeurs, les "réparer" pour que les publications fonctionnent à nouveau.

Universe.transf



Que ce soit en HTML ou en OD, le transformeur est toujours partiellement exact, avec juste une différence : **le titre est intégré dans les metas du sous modèle**, et non intégré à la part.



Placez vous dans le transformer web universe.transf contenu dans l'espace web,



La règles <for> peut s'appliquer à toutes les <parts> si on lui met comme **codes le **jocker****



Le titre présent dans chacune des parties n'est plus un **partTitle, mais un **subModelTitle****



En effet le titre n'est plus défini directement sous le `<sm:part>` mais dans la composition accessible via le `allowedModel` (qu'on appelle « sous-modèle »)

```
sm:for codes="*"
  sm:WHeadingBlock widgetClass=" " crossRefEntries=" " class="box "
    sm:title
      sm:subModelTitle prefix=" " suffix=" " format=" " axis=" "
      <sm:partTitle xmlns:sm="http://www.utc.fr/ics/scenari/v3/modeling"/>
      sm:fixedTitle value=" "
    sm:callSubModel axis=" "
  sm:defaultStyle ...
```

Écran 106



Les commentaires

Comme dans SCENARIchain, il est possible de désactiver des lignes en les passant en commentaire par *clic droit -> passer en commentaire*. Les lignes en commentaire sont alors affichées en vert dans l'éditeur.

Navigation vs Content



Mode "navigation" et mode "content"

Lorsque vous créez un transformeur (🔧 universe.transf) pour faire de l'HTML, il y a 2 possibilités principales de contexte :

- **Navigation** : vous n'avez pas encore créé la page, vous définissez l'arborescence de votre site en `<folder>` et `<pages>`
- **Content** : *votre page existe* et vous allez maintenant rajouter du contenu dans les transformeurs

universe.model (et **helloworld.model**) est notre item racine de la publication et nous devons commencer en mode `<navigation>`. Nous y avons défini une page, et n'en définirons aucune autre à l'avenir, donc dans les transformeurs des submodels, nous devons utiliser le mode `<content>`.

**sm:page**

<sm:page> permet de **créer physiquement** les fichiers html qui constitueront le site web.

astrobj.transf**astrobj.transf**

Le transformer pour **astrobj.model** pourrait être réalisé de manière très simple :

```
<for code="*" > WHeadingBlock > (title > subModelTitle) | callSubModel</pre>
```

Mais nous voulons que chaque partie "Atmosphère", "Surface"... ait un titre par défaut si l'auteur ne le précise pas.



créez ce nouveau transformer en faisant un clic droit dans l'atelier **helloworld sur l'espace **web** puis choisissez *Ajouter un item* et sélectionnez l'item de type *Publication* -> *Générateurs 'web'* -> *Transformers Web* -> *compositionXhtmlTransf* que vous nommerez **astrobj.transf**.**



Faites ensuite les modifications suivantes :



```
sm:compositionXhtmlTransf info=""
sm:model src:refUri="astrobj.model (/hellouniverse/model)"
sm:axis code=""
sm:fullTitleText ...
sm:shortTitleText ...
sm:content format="xhtml"
  sm:for codes="intro"
    sm:WHeadingBlock widgetClass="" crossRefEntries="" class="box"
    sm:title
      sm:subModelTitle prefix="" suffix="" format="" axis=""
      sm:fixedTitle value="Introduction"
      sm:callSubModel axis=""
    sm:defaultStyle ...
  sm:for codes="atm"
    sm:WHeadingBlock widgetClass="" crossRefEntries="" class="box"
    sm:title
      sm:subModelTitle prefix="" suffix="" format="" axis=""
      sm:fixedTitle value="Atmosphère"
      sm:callSubModel axis=""
    sm:defaultStyle ...
  sm:for codes="surf"
    sm:WHeadingBlock widgetClass="" crossRefEntries="" class="box"
    sm:title
      sm:subModelTitle prefix="" suffix="" format="" axis=""
      sm:fixedTitle value="Surface"
      sm:callSubModel axis=""
    sm:defaultStyle ...
  sm:for codes="core"
    sm:WHeadingBlock widgetClass="" crossRefEntries="" class="box"
    sm:title
      sm:subModelTitle prefix="" suffix="" format="" axis=""
      sm:fixedTitle value="Noyau"
      sm:callSubModel axis=""
    sm:defaultStyle ...
  sm:for codes="part"
    sm:WHeadingBlock widgetClass="" crossRefEntries="" class="box"
    sm:title
      sm:subModelTitle prefix="" suffix="" format="" axis=""
      sm:fixedTitle value=""
      sm:callSubModel axis=""
    sm:defaultStyle ...
```

Écran 107



Remarque

On **sélectionne** avec plusieurs `<for>` **chaque part une par une grâce au code**. Lorsque vous précisez plusieurs `<title>` à la suite l'un de l'autre, **c'est le premier existant qui est choisi** : si le `<subModelTitle>` existe, le titre de l'auteur sera utilisé. Pour les parties libres nous ne leur donnons pas de titre si l'auteur ne le spécifie pas.

part.transf et déclarations de transformers





Cette fois-ci, c'est plus simple : il s'agit juste de transférer le travail de transformation à la primitive texte : `<for code="*" > callSubModel >` :

```
sm:compositionXhtmlTransf info=" "  
  sm:model sc:refUri="part.model (/helloUniverse/model)"  
  sm:axis code=" "  
  sm:fullTitleText ...  
  sm:shortTitleText ...  
  ?  
  sm:content format="xhtml"  
    sm:for codes="*" "  
      sm:callSubModel axis=" "
```

Écran 108

Écran de contrôle

Salut l'univers (Définition de l'atelier wsp)

Édition Propriétés Contrôles

Models
Generator huOd : Générateur OpenDoc...
Generator huWeb : Générateur XHTML

huWeb : Générateur XHTML

Transformers in use

- 1 uiFrame
- 2 navigation hw_universe universe.transf
- 3 title hw_astrobj No transformer
- 4 xhtmlContent hw_astrobj No transformer

Transformers not used

Transformers never called

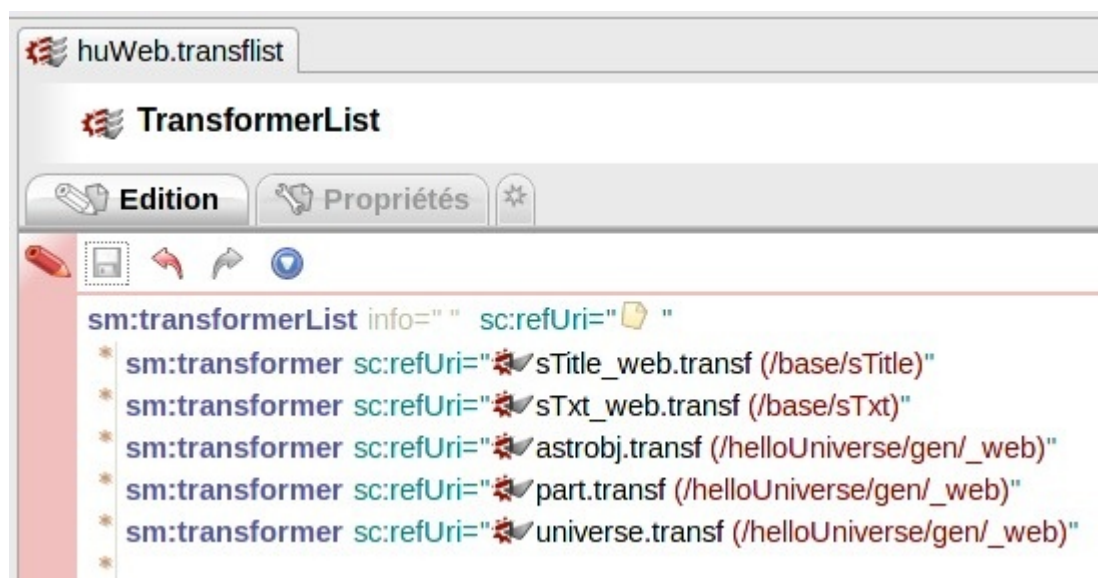
- sTxt_web.transf
- sTitle_web.transf

Écran 109



Mettez à jour les transflists

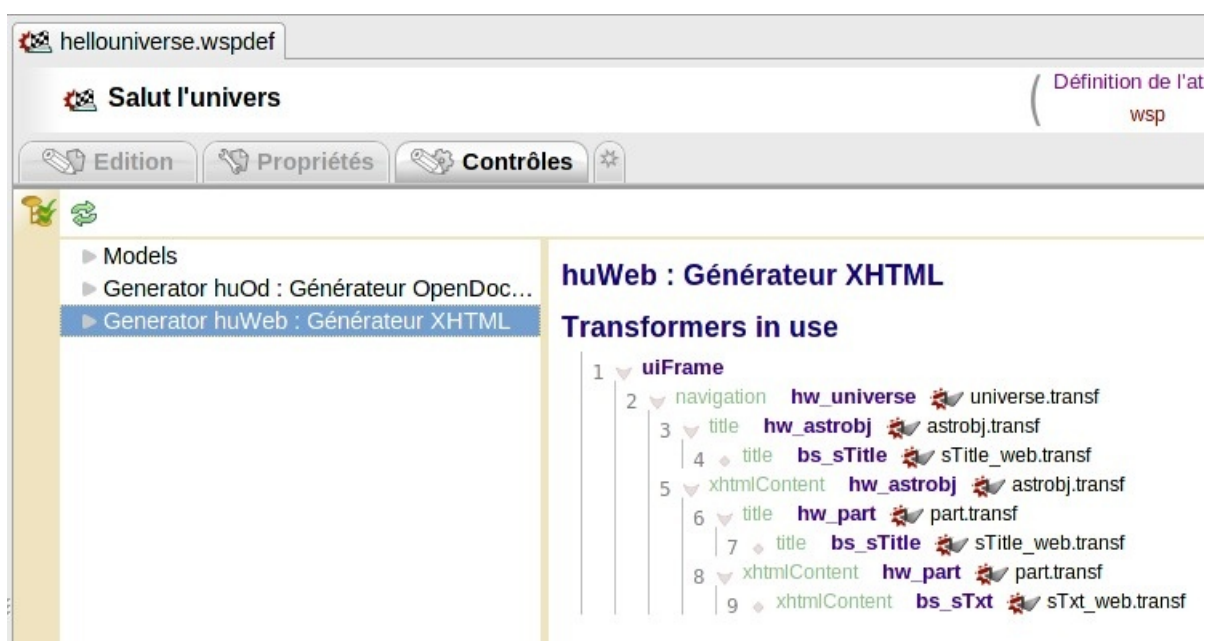
Mettez à jour le fichier **.transflist** du générateur HTML, ajoutez-y les 2 nouveaux transformers.



Écran 110

Écran de contrôle

Il n'y a plus d'erreur...



Écran 111

En cas d'erreur...



checklist

En cas d'erreur, pensez toujours à vérifier :

- Les écrans de contrôles ;
- La **correspondance entre les codes** des **.model** et **.transf** ;
- Que les codes des parts et les codes des **.models** sont **uniques** ;
- Que vos TransfList sont complètes ;
- Que vous n'avez pas mis de transformers OD dans un générateur HTML ou inversement ;
- Que vous avez fait les bons choix entre **navigation** ou **content** dans les XhtmlTransf (l'écran de contrôle peut vous aider sur ce point) ;
- Que vous n'avez pas mélangé les appels : entre **callSubModelTitle** et **callPartTitle** par exemple ;
- Que vous n'avez **pas d'accent** dans les noms de fichier : les accents sont nuisibles et sources d'incompatibilité entre les plateformes (conseil valable pour toutes les applications Scenari 3.7) ;
- Que votre modèle ne comporte pas de nom de code, de fichier ou d'axis anormalement long : si la **longueur totale du chemin** d'un fichier dépasse 255 caractères, cela peut créer des problèmes sous Windows. SCENARibuilder crée des sous répertoires de travail avec les codes que vous utilisez dans votre modèle, restez donc raisonnable ;
- Que vous n'avez pas oublié d'**enregistrer vos modifications** avant de lancer vos nouvelles compilations.

2.3. Publication OD



Truc & astuce

Pour l'OpenDocument Text, il n'y a aucune difficulté supplémentaire, vous devez faire les mêmes changements ou nouveaux transformers



Mettez à jour **universe.transf** en utilisant un **for** par **part**, en prenant garde à la construction des titres.



Créez et paramétrez **part.transf**



Attention

Vous ne pouvez pas copier le paramétrage d'un transformeur OD pour le coller dans un transformeur XHTML, les logiques de publications étant différentes, les paramétrages peuvent eux aussi différer.



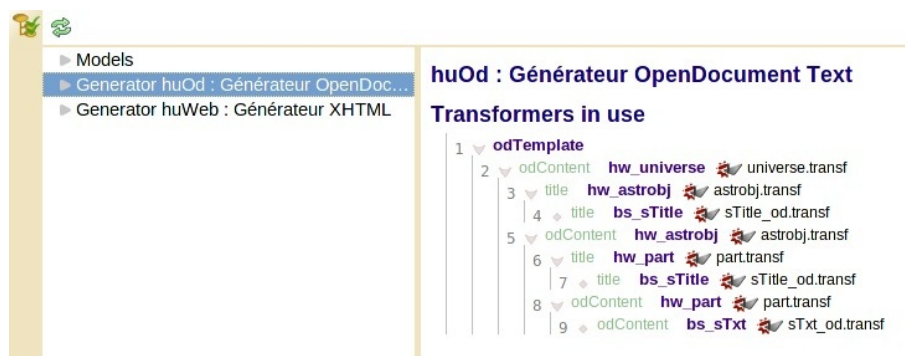
Créez et paramétrez **astrobj.transf**, en utilisant un **for** par **part**, en prenant garde à la construction des titres.



Déclarez les transformeurs dans le translist du générateur OD



Vérifiez via l'écran de contrôle que votre paramétrage est bon.



Écran 112



Compilez un SCENARIttest et testez le générateur OD



3 Les données de type "formulaire"

3.1. Présentation et modélisation

Comme vous avez pu le voir dans l'analyse, les planètes ont aussi des données précises : nom, diamètre, masse, que nous qualifierons de **méta-données**. Ce type d'information n'est pas bien adapté à une structure type "bloc de contenu" ou "ensemble de paragraphes" car trop peu contrainte. Nous n'utiliserons donc pas les `<compositionPrim>`, mais un nouveau type de primitive de contenu : les `<dataformPrim>`.



dataForm

Permet de créer des modèles de champs de données structurées. Se rapproche de la représentation d'un formulaire.

Quand utiliser des dataforms ?

La primitive dataform est très importante pour une modélisation fine des documents : contrainte de champs courts, ou de typage précis (nombre, date) ; ou encore pour ajouter des paramètres de publications (pour les modélisateurs expérimentés uniquement).



Intégration des dataforms

Vous avez déjà fait usage de méta-données : le titre est une meta donnée avec une seule information.



Dans **astrobj.model**, vous allez supprimer la référence à **sTitle.model**



Attention

Supprimer la référence \neq Supprimer l'item



à la place, créez un nouvel item : dans l'écran de création des items : **Primitives** -> **Primitives de métadonnées** -> **dataFormPrim.** de nom **astrobjMeta.model** ou **astrobjM.model**



Renseignez les champs namespace, prefixe et code



créez le premier **<field>** (*champ de données*) de la dataform : le nom de l'astre.



Ajoutez un champ Photo de type **refItem** autorisant l'insertion d'une image de type jpeg



Truc & astuce

Les formats d'images sont disponibles dans l'espace **binaries**



Mettez les modèles ainsi renseignés dans le **wspdef** dans le **<publicClasses>** pour que l'auteur puisse les créer.

```
sm:wspDefinition keyWsp="universe" name="universe" defaultWspCode=" "
majorVersion="0" mediumVersion="0" minorVersion="1"

sm:publicClasses
  sm:class sc:refUri="universe.model (/helloUniverse/model)" access="readWrite"
    sm:mainView
      sm:formEditorTab name="Edition" display="visible" formEditorKey=" "
      sm:blocksTab name="Publication" display="visible"
      sm:generatorsBlock name="Publication" displayMode="lockedOpen"
        sm:genBox
          sm:generator sc:refUri="huOd.generator (/helloUniverse/gen/od)"
          sm:deployment ...
        sm:genBox
          sm:generator sc:refUri="huWeb.generator (/helloUniverse/gen/web)"
          sm:deployment ...
    sm:defaultMainView ...
    sm:wspUi ...
    sm:overlays ...
    sm:migration ...

  sm:class sc:refUri="astrobj.model (/helloUniverse/model)" access="readWrite"
  sm:group name="Images"
    sm:class sc:refUri="jpeg.model (/binaries/jpeg)" access="readWrite"
    sm:class sc:refUri="png.model (/binaries/png)" access="readWrite"
```

Écran 113



Nous souhaitons maintenant demander à l'auteur le type de planète, ce type va toujours appartenir à un ensemble fini de possibilités : "étoile", "tellurique" ou "chthonienne". Créez un champ de type **enum**



Pour les satellites qui contiennent 2 données : le nom et la distance, l'approche la plus intuitive serait de créer une autre dataform et au lieu de déclarer un nouveau <field>, de faire un <subData>. Cette approche est tout à fait valable, mais comme nos données sont relativement simples, nous pouvons plutôt faire un <group> "satellite" et ajouter à l'intérieur 2 <field> "nom" et "distance".



Recopiez les éléments suivants dans le fichier **astrobjMeta.model**:

```
sm:dataFormPrim name="Info/astre" info=" "
sm:help ...
sm:identification targetNamespace="scenari-platform.org:helloworld" targetPrefix="hw" code="astrobjMeta"
sm:structure
  * sm:field code="name" name="Nom" occurrence="one" family=" "
    sm:help ...
    ? sm:string maxCharacters=" "
  * sm:field code="photo" name="Photo" occurrence="zeroOrOne" family=" "
    sm:help ...
    ? sm:refItem
      * sm:allowedModel sc:refUri="jpeg.model (/binaries/jpeg)"
      * sm:allowedModel sc:refUri="png.model (/binaries/png)"
  * sm:field code="nat" name="Nature" occurrence="one" family=" "
    sm:help ...
    ? sm:enum
      * sm:option key="star" name="Etoile"
      * sm:option key="tel" name="Planète tellurique"
      * sm:option key="cht" name="Planète chthonienne"
  * sm:group code="sat" name="Satellite" occurrence="zeroOrMore" family=" "
    sm:help ...
    * sm:field code="name" name="Nom" occurrence="one" family=" "
      sm:help ...
      ? sm:string maxCharacters=" "
    * sm:field code="distance" name="Distance" occurrence="one" family=" "
      sm:help ...
      ? sm:string maxCharacters=" "
    * sm:field code="dia" name="Diamètre" occurrence="zeroOrOne" family=" "
      sm:help ...
      ? sm:string maxCharacters=" "
    * sm:field code="mass" name="Masse" occurrence="zeroOrOne" family=" "
```

Écran 114



Nous avons vu que les dataforms sont une liste de champs. Lorsque le contexte s'y prête, vous devez choisir un des champs de la dataform qui sera utilisé en tant



que titre. Dans notre exemple, c'est le champ ayant pour code **name** (le nom de l'astre) qui sera utilisé comme titre.



Titrez vos meta

Dans le fichier **astrobjMeta.model**, allez au niveau de la balise <storage> puis double-cliquez sur la balise <fullTitleBuilder> et taper **name** dans refCode.

```
sm:storage
  sm:fullTitleBuilder
    sm:fieldValue refCode="name" prefixIfExist=" " suffixIfExist=" "
  sm:shortTitleBuilder returnFullTitleIfEmpty="no"
    sm:fieldValue refCode=" " prefixIfExist=" " suffixIfExist=" "
```

Écran 115

Jusqu'à présent nous avons uniquement parlé de "titres simples". Dans certains cas, nous pourrions être amené à demander à l'auteur 2 types de titres, un **fullTitle** et un **shortTitle**, pour pouvoir soit être exhaustif lorsque l'on dispose d'une ligne complète pour publier le titre, soit de faire un petit lien de menu ou synthétiser encore plus lorsque l'on doit *faire tenir ce titre en quelques caractères*. Pour utiliser ce genre de "double titre", insérer dans le fichier **part.model** soit le fichier **title.model** des modelets à la place de <sTitle>, soit une dataform dans laquelle vous allez définir les 2 mécanismes <fullTitleBuilder> et <shortTitleBuilder>.

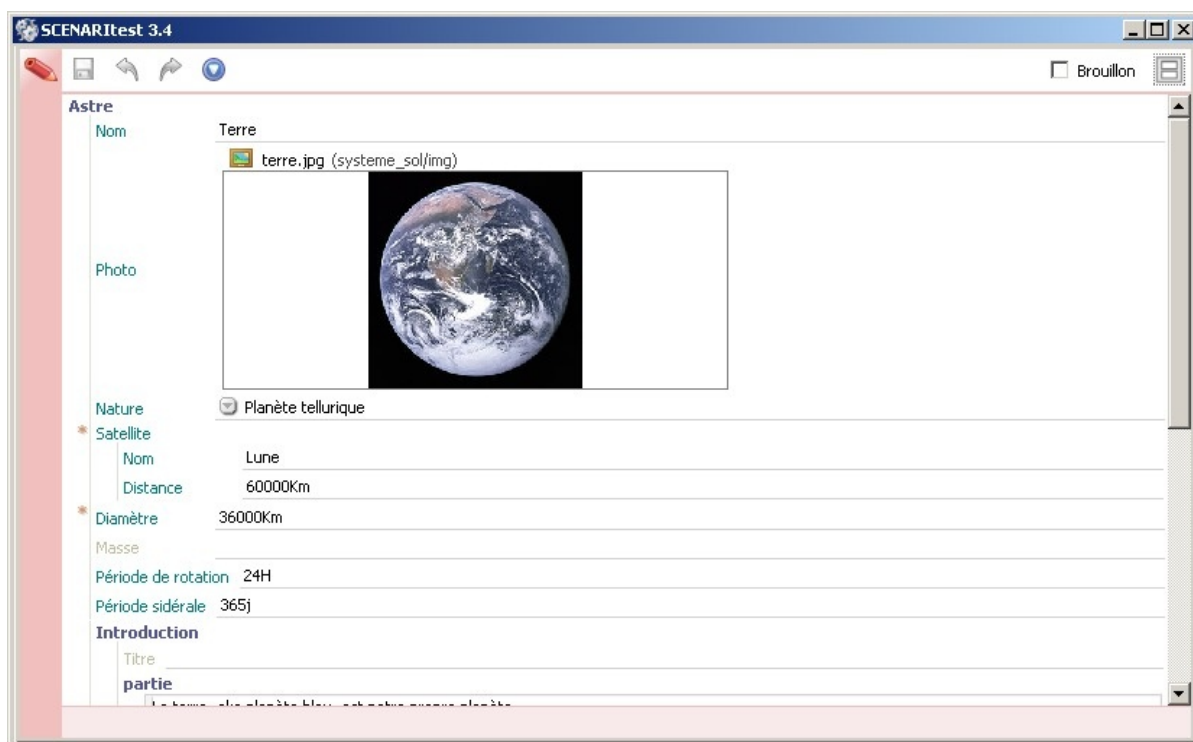


Autres points de départs pour les dataforms

Il existe au moins 3 méthodes pour intégrer des dataforms :

- sm:meta : rattaché à l'ensemble de la composition (comme c'est le cas ici)
- sm:part -> sm:meta : rattaché uniquement à une part
- subdata : à l'intérieur d'une dataform, permet de pointer vers une autre dataform.

Résultat dans SCENARitest



Écran 116



Construction du titre

La construction du titre est une partie très importante de la modélisation. Une dataform pouvant contenir tout un tas de champs, il est absolument nécessaire d'indiquer explicitement le ou lesquels de ces champs peuvent constituer le titre.

Ce n'est pas parce qu'on nomme un champ title ou name que le titre sera automatiquement l'un de ces champs.

3.2. Publication HTML

Il est courant de vouloir afficher une série d'informations de types formulaire sous la forme d'un tableau synthétique. C'est pourquoi nous allons utiliser cette approche pour le tutoriel.





Préparez sur papier une maquette du tableau qui va afficher toutes les informations d'un astre.



Truc & astuce

Cela vous permettra de mieux visualiser comment paramétrer le transformer.



faites un clic droit sur votre espace où vous avez rangés les transformers web (hellouniverse/gen/_web)



sélectionnez *Ajouter un nouvel item* et choisissez *Publication* -> *Générateurs 'web'* -> *dataFormXhtmlTransf* nommé comme le .model de notre dataform : **astrobjM.transf** (ou **astrobjMeta.transf**)



Utiliser les widgets **WTable**, **WTableRow** et **WTableCell** pour construire un tableau

Le **WTable** est un tableau "libre", le modélisateur va créer des lignes + des colonnes et les remplir. Il est adapté à des tableaux à contenus variés ou à des tableaux en largeur. Nous allons créer un tableau à 2 cellules : à gauche, la photo et à droite, les autres informations de la dataform. Un <WTable> contient des <WTableRow> puis des <WTableCell>.



Pour afficher le contenu d'un champ (**field**), utilisez **forField**, avec le code du champ à récupérer, et à l'intérieur **fieldValueXXX** selon que le champ à afficher est du type **String**, **Date**, **RefItem**.



Vous pouvez aussi utiliser **W2ColsFreeRow** pour créer un tableau pré paramétré



Exemple

```
sm:dataFormXhtmlTransf info=" "  
  sm:model sc:refUri="❌ astrobjMeta.model (/hellouniverse/model)"  
  sm:axis code=" "  
  sm:fullTitleText ...  
  sm:shortTitleText ...  
  ?  
  sm:content  
    * sm:WTable widgetClass="astrobjMeta1" crossRefEntries=" "  
      * sm:col label=" "  
      * sm:col label=" "  
      * sm:WTableRow widgetClass=" " crossRefEntries=" "  
        * sm:WTableCell widgetClass="astrobjMeta1" crossRefEntries=" "  
          * sm:forField codes="photo"  
            * sm:fieldValueRefItem  
              sm:callRefModel axis=" "  
          *  
        *  
      * sm:WTableCell widgetClass="astrobjMeta2" crossRefEntries=" "  
        * sm:W2Cols widgetClass=" " crossRefEntries=" "  
          * sm:forField codes="nat"  
            * sm:W2ColsFreeRow widgetClass=" " crossRefEntries=" "  
              * sm:labelCell  
                * sm:fieldName  
                *  
              * sm:valueCell  
                * sm:fieldValueEnum format="📄 name"  
                *  
              * sm:defaultStyle ...  
            * sm:forField codes="dia mass rot sid"  
              * sm:W2ColsFreeRow widgetClass=" " crossRefEntries=" "  
                * sm:labelCell  
                  * sm:fieldName  
                  *  
                * sm:valueCell  
                  * sm:fieldValueString  
                  *  
                * sm:defaultStyle ...  
              * sm:defaultStyle ...  
            * sm:defaultStyle ...  
          * sm:defaultStyle ...  
        *  
      * sm:defaultStyle ...  
    * sm:defaultStyle ...
```

Écran 117



Modélisation avancée

Les conditions d'existence : Choose / When

Dans ce tutoriel, nous voulons, si une planète a un ou des satellites, ajouter un bloc avec pour titre "satellites" dans notre publication HTML, mais que cette partie n'apparaisse pas dans le cas contraire.

Nous utilisons pour cela l'ensemble `<choose>` / `<when>` / `<otherwise>`. La condition est déterminée dans le `<when>` : elle peut être l'existence d'un code dans les données (donc, qu'une `<part>`, qu'un `<field>` ou qu'un `<group>` existe), avec l'attribut `<codes>`, si les codes existent ce qui est dans le `<when>` est appliqué, dans le cas contraire ce qui est dans le `<otherwise>` est appliqué. `<When>` ne sélectionne pas les données, il faut donc refaire un `<for>` après. Si nous réécrivons le transformer de la même manière sans `choose` / `when`, le titre "satellites" sera publié même pour les parties n'en ayant pas.

```
sm:choose
  sm:when codes="sat" sc:refUri=" " xpath=" "
    sm:WBlock widgetClass="satTitle" crossRefEntries=" " tagName=" "
      sm:WText value="Satellites"
    sm:WBlock widgetClass="satBlock" crossRefEntries=" " tagName=" "
      sm:forGroup codes="sat"
        sm:WBlock widgetClass="satLine" crossRefEntries=" " tagName=" "
          sm:forField codes="name"
            sm:WBlock widgetClass="name" crossRefEntries=" " tagName="span"
              sm:fieldValueString
          sm:forField codes="dist"
            sm:WBlock widgetClass="dist" crossRefEntries=" " tagName="span"
              sm:WText value=":"
              sm:fieldValueString
            sm:otherwise
```

Écran 118



Des conditions plus complexes avec xpath

À l'aide d'expressions xpath, nous avons plus de liberté pour poser des conditions sur le contenu des champs, sur le nombre d'éléments... Exemple de condition XPATH sur la valeur d'un champ enum.



Conserver la structure et la hiérarchie

Pour publier la liste des satellites, nous avons 3 balises de structure dans le **dataform.model** :

- Le groupe **sat**
- En sous niveau de **sat**, le champ **name** (nom du satellite)
- En sous niveau de **sat**, le champ **dist** (distance)

Il est très important de **conserver la même hiérarchie** : commencer par un <forGroup> puis des <forField>. Pour cette partie, au lieu de tableaux, nous utilisons des <WBlock> qui se traduisent dans le HTML par des <div> ou des .

« Appel des métas »



Depuis le transformer `astrobj.transf`, faites un appel au transformer de ses métadonnées (`callCompositionMeta`) afin d'exécuter la règle qui affiche les métadonnées sous forme de tableau.

Pensez à déclarer ce nouveau transformer.



Écran de contrôle

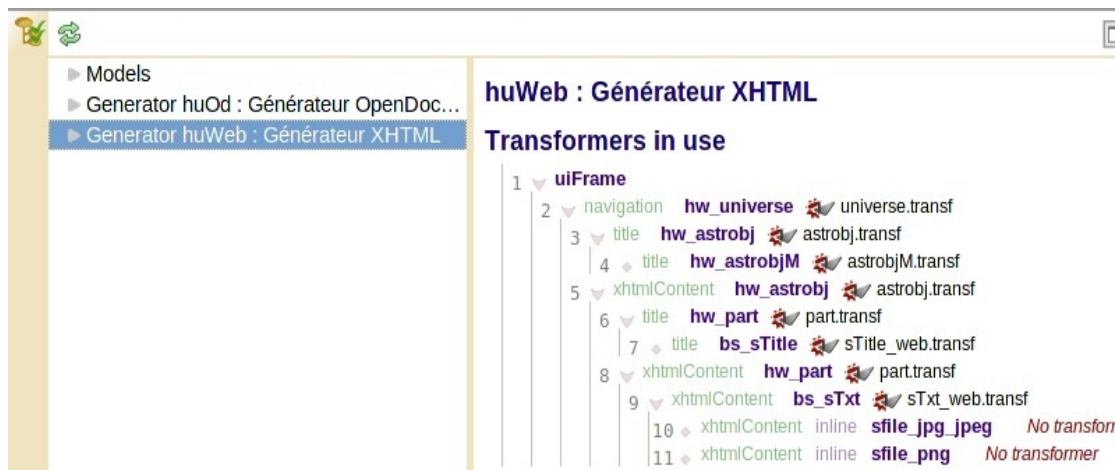


Conseil

N'oubliez pas de passer par l'écran de contrôle pour vérifier votre paramétrage.



Exemple



Écran 119

Ici nous voyons que nous avons oublié de déclarer les transformers d'images.

3.3. Publication OD

Pour créer la version OD du transformer, vous pouvez adopter approximativement la même méthode.



WParagraphe

Une chaîne de caractère, un nombre ou une date doivent impérativement être intégrés dans un paragraphe afin de respecter la norme OpenDocument.



Limites physiques de la page

- Il est très déconseillé de proposer à l'auteur de rentrer des paragraphes longs dans des tableaux, la gestion des tableaux étendus sur plusieurs pages peut poser problème lors de l'ouverture du document avec OpenOffice.
- Il est déconseillé d'imbriquer les tableaux ou les WBox pour les mêmes raisons.



4 Pages de la publication

La notion principale de **page** diffère en fonction du contexte technique :

- sur un support papier, la page est délimitée physiquement, en A4 par exemple ;
- sur le web, la page est associée à un lien puis à une fenêtre, dont le contenu n'a pas de limite physique, un ascenseur permettant de « descendre ».

Mais on peut lui donner un caractère sémantique :

- la page d'accueil ;
- la table des matières, le plan ;
- un complément en annexe ou en sur-fenêtre ;
- etc.

Une nouvelle page va créer pour le lecteur une rupture dans sa lecture, et ainsi lui indiquer une rupture logique (il n'y a plus de place) ou sémantique (je suis sur un nouveau chapitre, sur le glossaire etc).

4.1. HTML : menu et hiérarchie de pages



Conseil

Nous partons d'un document composé uniquement d'**une seule page**. Tout le contenu est à l'intérieur de cette page. Nous allons chercher à faire un document en plusieurs pages HTML : une page par planète, avec un menu **pour naviguer entre les planètes**.

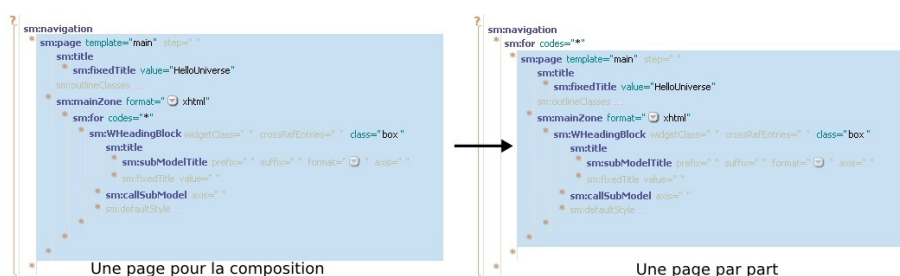
Création de la page au niveau des planètes



Pour tenir notre cahier des charges, nous devons modifier l'emplacement de la création de page : passer de une page pour la composition **universe.model** à une page par part de cette composition.



Dans votre espace **web**, ouvrez l'item **universe.transf** et modifiez les données suivantes :



Écran 120



Solution alternative : nous aurions pu juste faire un `<<for codes="*"><callSubModel>>` dans `universe.transf` pour décharger ce transformer de la création des pages, puis la reporter dans `astrobj.transf`, qui dans ce cas doit être en mode `<navigation>` et non `<content>`.

Création du menu



Nous avons défini les pages, mais il n'y a aucun moyen d'accès pour les choisir. Nous allons donc déclarer un menu (ou outline) dans `huWeb.uiTemplate`.



On crée une div `menu` qui sera positionnée sur la page lors de l'étape de stylage.



Cette div contient un widget de type `outlineUiWidget`. (utilisez l'astérisque pour créer la balise `Widget` et créez un item de type `OutlineUiWidget`)

```
sm:body
*
div id="menu" class=" "
*
sm:Widget sc:refUri="webOutline.uiwidget"
*
div id="content" class=" "
*
sm:callMainZone format="xhtml"
*
```

Écran 121



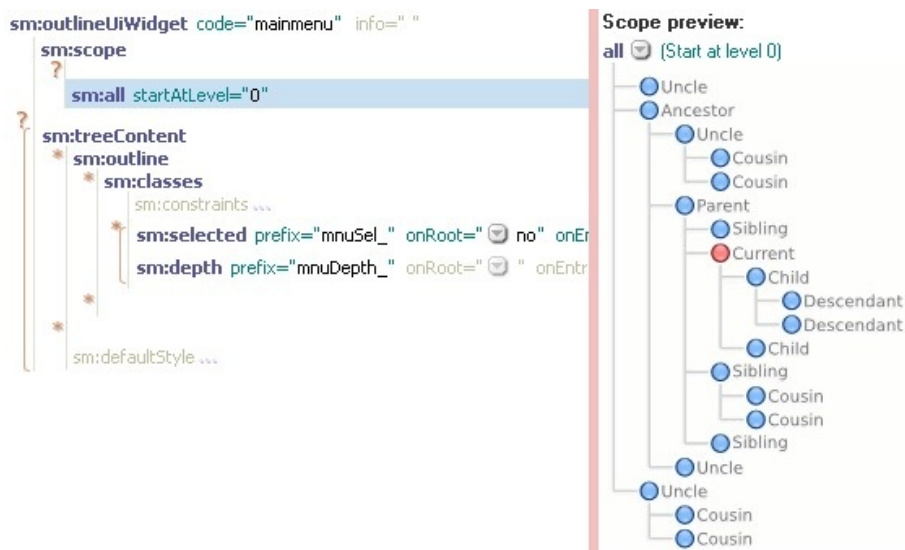
Nous allons modifier 2 paramètres de ce widget : le scope : all

le scope : Les éléments visibles ou non dans le menu. Un scope "all" dévoilera tous les éléments du menu (Sélectionnez le et consultez les différentes possibilités dans la partie droite de la fenêtre : liste de toutes les pages parentes, liste des fils...).



Les classes

Permettent de choisir des styles CSS à appliquer dans les éléments du menu, en fonction de conditions précises (position dans l'arbre par rapport à la page courante, profondeur, type d'éléments...).



Écran 122



```
sm:compositionXhtmlTransf info="" "  
  sm:model sc:refUri="🔗 universe.model (/hellouniverse/model)"  
  sm:axis code=""  
  sm:fullTitleText ...  
  sm:shortTitleText ...  
  sm:navigation  
    sm:page template="main" step="" "  
      sm:title  
        sm:fixedTitle value="Introduction"  
        sm:outlineClasses ...  
        sm:mainZone format="📄 xhtml"  
          sm:WFreeBlock  
            div class="welcome"  
              Bienvenu dans l'univers... Choisissez un astre dans le menu a votre gauche  
            sm:defaultStyle ...  
      sm:for codes="*" "  
        sm:page template="main" step="" "  
          sm:title  
            sm:subModelTitle prefix="" suffix="" format="📄 " axis="" "  
            sm:fixedTitle value="" "  
            sm:outlineClasses ...  
            sm:mainZone format="📄 xhtml"  
              sm:WHeadingBlock widgetClass="" crossRefEntries="" class="box "  
                sm:title  
                  sm:subModelTitle prefix="" suffix="" format="📄 " axis="" "  
                  sm:fixedTitle value="" "  
                  sm:callSubModel axis="" "  
                  sm:defaultStyle ...  
        sm:for codes="*" "
```

Écran 123



Résultat

Notre contenu n'ayant pas une structure extra-ordinaire, vous devriez simplement voir apparaître une liste des pages en début de votre publication HTML, et le contenu de la première sera affichée par défaut.



- Introduction
- [Astre1](#)
- [Astre 2](#)

Bienvenue dans l'univers... Choisissez un astre dans le menu à votre gauche.



Les folders

Pour créer une *hiérarchie de page* plus complexe, avec des sous-niveaux, on déclare des **Folders** et on y crée les pages à l'intérieur. L'option `<folderPreview>` est la page qui s'affiche lorsque l'utilisateur clique sur le dossier dans le menu, sans `<folderPreview>` c'est la première page de contenu du folder qui est utilisée.

4.2. ODT : Page de garde, table des matières, numérotation

Introduction

Nous allons améliorer notre publication ODT avec les fonctionnalités suivantes :

- Page de garde
- Table des matières
- Saut de page après chaque planète décrite



Page de garde

Créer la page de garde



Pour déclarer une page de garde, on crée un nouveau `<pagemaster>` dans **huOd.template**.



On y met un `<WflowTextArea>` : une zone dont la position est définie librement par le modélisateur, dans le fichier de styles

```
sm:contentStruct
  * sm:WPageMaster style="first " name=" "
    sm:pagesLayout ...
    sm:header ...
    sm:body
      * sm:WFlowTextArea style="title "
        * sm:fixedString value="L'univers"
      *
    sm:footer ...
```

Écran 124



re-générez le fichier de style



révélez le nouveau style généré



glissez-déposez le nouveau style à la place de l'ancien



Si vous voulez afficher le titre "L'univers" au centre de votre page de garde, ouvrez le fichier de styles et déplacez la nouvelle zone rectangulaire à sa bonne position sur la page. Cette opération se fait à la souris, sans passer par la fenêtre de paramétrage des styles.



Table des matières



Insérer un sommaire et le paramétrer

OpenOffice permet de générer automatiquement une table des matières à partir des différents textes de titre, dans le cas de documents Scenari, à partir des <WHeading>.



On déclare la table des matières dans le template :

```
sm:WPageMaster style="content " name=" "  
  sm:pagesLayout ...  
  sm:header ...  
  sm:body  
    sm:WTableOfContents style="toc "  
      sm:title  
        Table des matières  
    sm:callRootModel axis=" "
```

Écran 125



Mettez le style à jour



faites un clic droit sur l'item **huOd.skin.odt** puis choisissez *Ouvrir dans l'éditeur par défaut du système*.

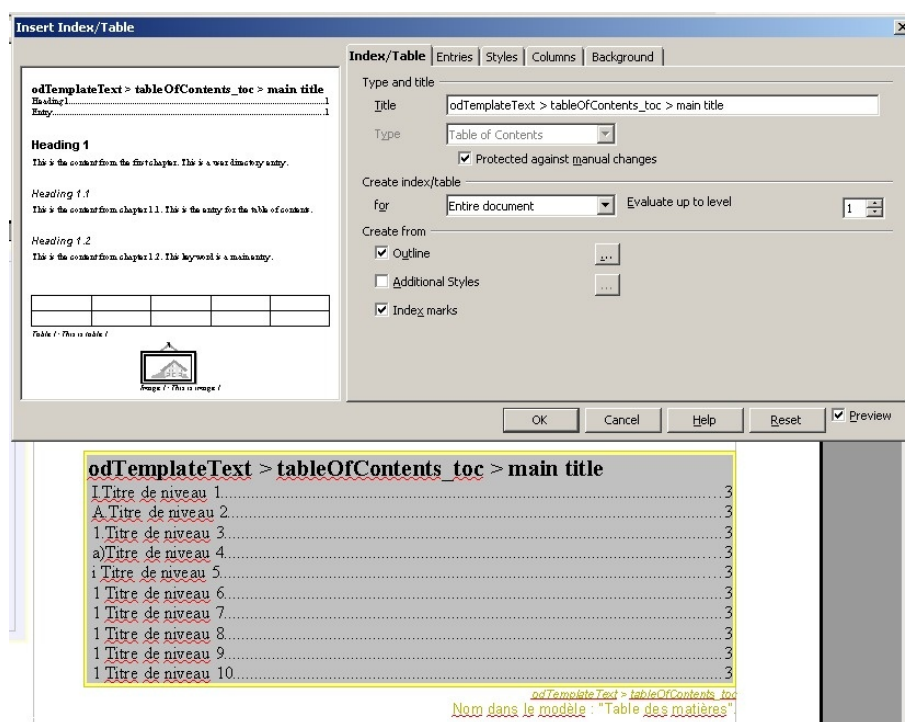
Le fichier s'ouvre sous OpenOffice ou LibreOffice



allez dans le menu *Insert -> Indexes and Tables -> Indexes and Tables*,




on obtient cet écran en simulant l'insertion d'une nouvelle table des matières à l'intérieur de celle déjà existante. Dans notre cas, nous allons uniquement mettre le nom de l'astre et non les titres des sous-parties, on règle donc l'option **Evaluate up to level** à 1.



Écran 126

Numérotation des pages

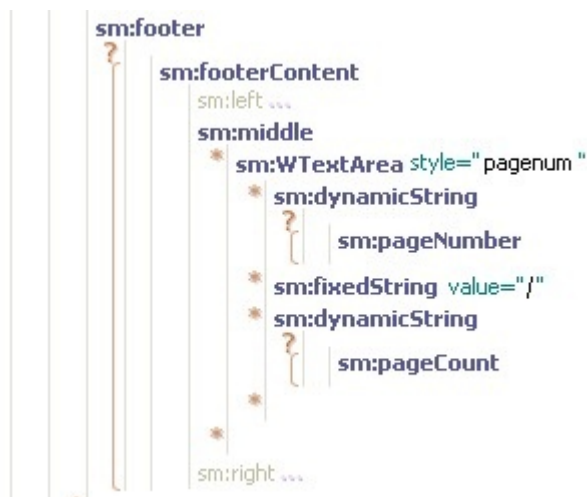
Toujours dans le  template, on souhaite ajouter un pied de page au format "numéro de page" / "nombre total de pages du document" :



Se placer au niveau du PageMaster **content**



Renseignez le footer avec un WTextArea et deux dynamicString pour le numéro de page et le nombre de page.



Écran 127



Mettez le style à jour

Compléments et exemple

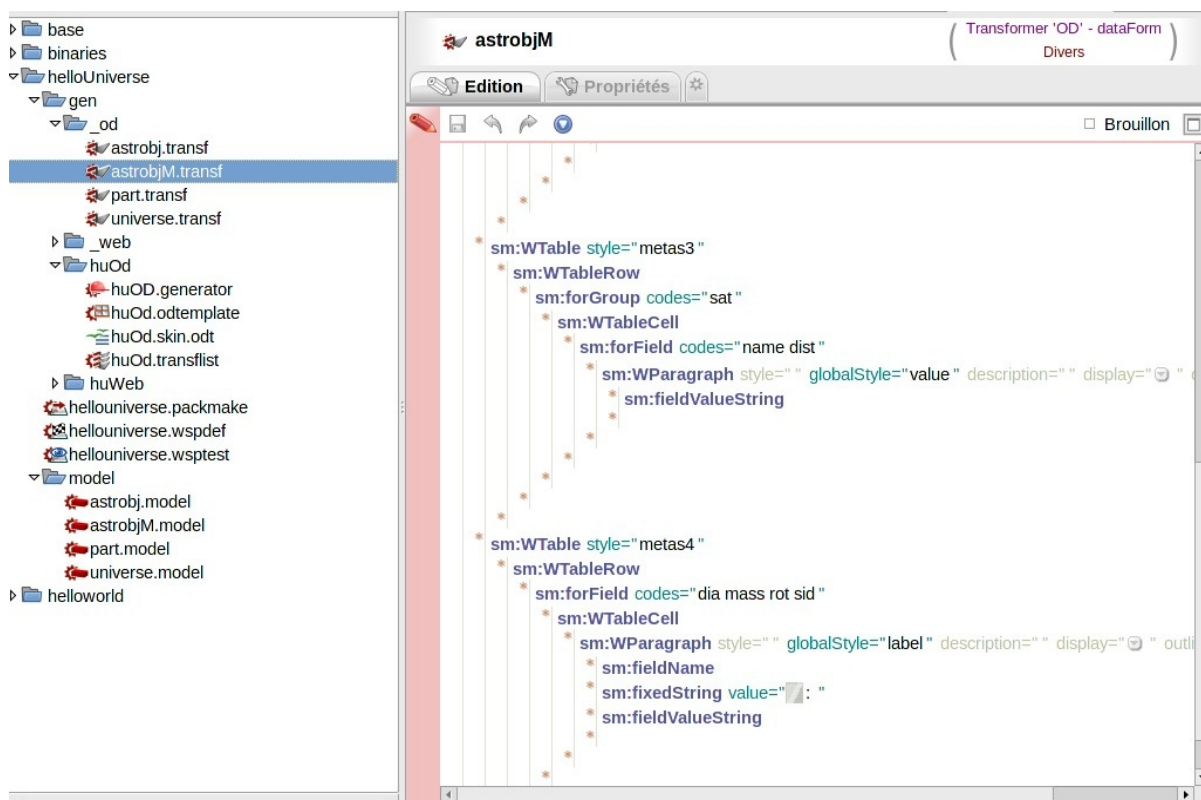


Sauts de pages entre les chapitres

Les sauts de pages entre les chapitres se paramètrent dans le fichier de styles. Vous pouvez appliquer la règle « saut / de page / avant » sur le style de paragraphe correspondant au titre de niveau 1.



Exemple



5 Les axis

Problématique

Pour publier un document, nous avons souvent besoin de publier une donnée plusieurs fois, et de plusieurs manières différentes. Lorsque l'on fait un appel classique à un transformer, **c'est toujours le même qui est appelé**, et il publie toujours les données **de la même manière**, avec toujours les mêmes classes pour le stylage.



Axis

Le rôle des axis est de définir *plusieurs versions d'un même transformer*, qui peuvent publier la même donnée de plusieurs manières différentes. Autrement dit : Un axis est une variante d'un transformer, il permet de parcourir un .model dans un **contexte** particulier.



Exemple

Par exemple, dans notre liste de planètes, nous pouvons rajouter une page "galerie" qui va afficher des miniatures de toutes les photos des planètes.



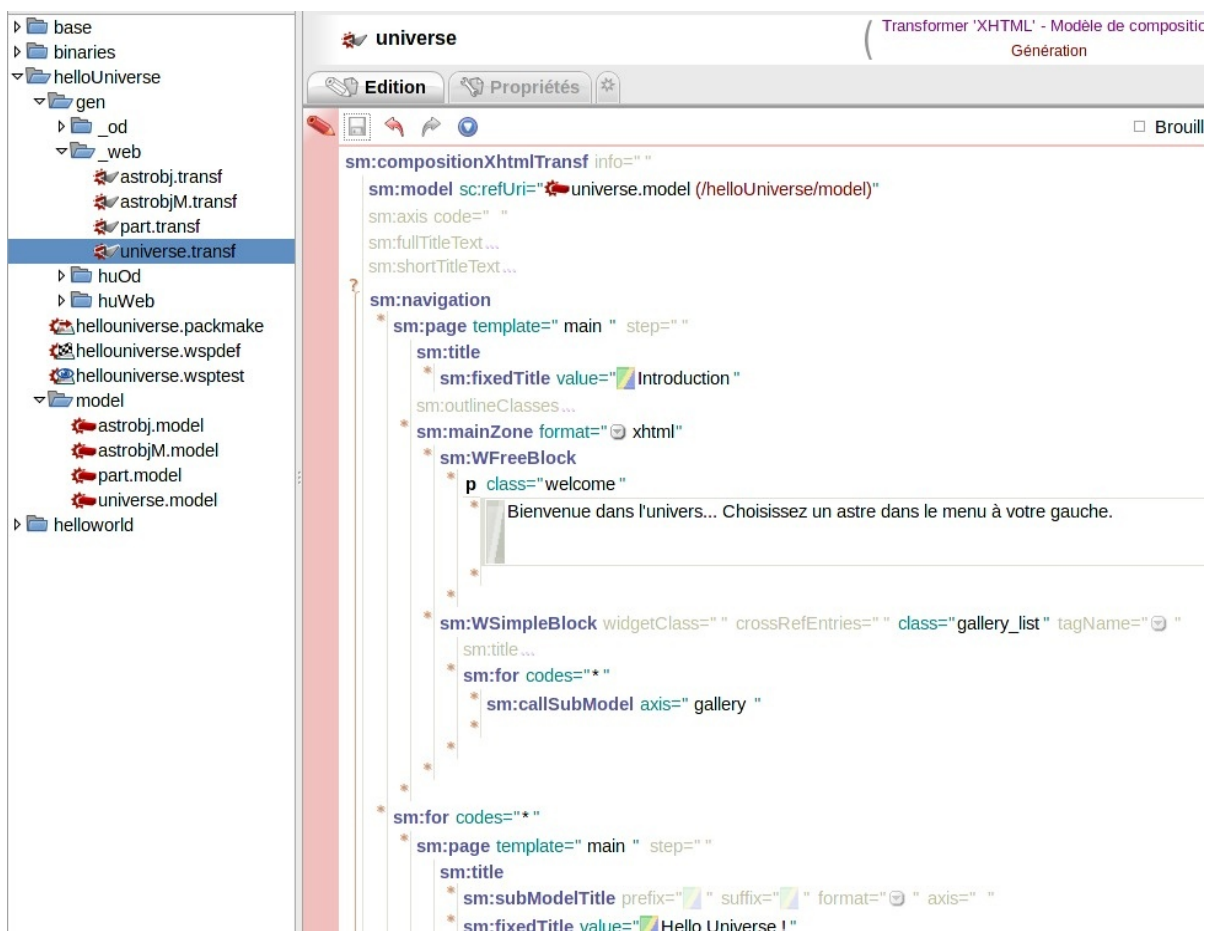
Remarque

En XSL, ce concept se rapproche de celui de l'attribut **mode** sur `xsl:apply-template` et sur `xsl:template`.



Appeler des transformers avec axis

À partir d'un transformeur ou du template, lorsque l'on utilise `<callSubModel>`, `<callRootModel>`, `<callSubMeta>`, `<callMeta>`, ou autre appel à un transformeur fils, nous avons la possibilité de préciser un axis. Par exemple, dans **universe.transf**, on rajoute un appel, pour chaque astre de la liste, avec l'axis **gallery**.



Écran 128



La hiérarchie des transformers avec axis

Vous allez maintenant créer les transformers fils pour reproduire toute une hiérarchie jusqu'à arriver au niveau "image".



Conseil

Par convention, on nomme les fichiers axis avec transformers sous la forme **codeObjet@codeAxis.transf**.



Le premier `<callSubModel axis gallery>` situé dans **universe.transf** est un appel au transformer de **XXXX.model** d'axis **YYYY**. Dupliquez **XXXX.transf** en **XXXX@YYYY.transf**



Paramétrez le transformer en renseignant bien l'axis dans la définition du transformer (sous `sm:model`), et faites un appel à ses métadonnées d'axis gallery.

On cherche à afficher seulement quelques informations sur l'astre, donc on ne veut pas toutes les métadonnées, seulement une sélection.



Notre deuxième appel `<callCompositionMeta axis gallery>` que l'on vient de paramétrer est un appel au transformer de **XXXX.model** d'axis **YYYY**. Dupliquez **XXXX.transf** en **XXXX@YYYY.transf**



Paramétrez le transformer en renseignant bien l'axis.

Proposition : afficher seulement la photo en miniature (avec encore une fois un axis gallery pour paramétrer l'affichage de l'image dans ce contexte précis).



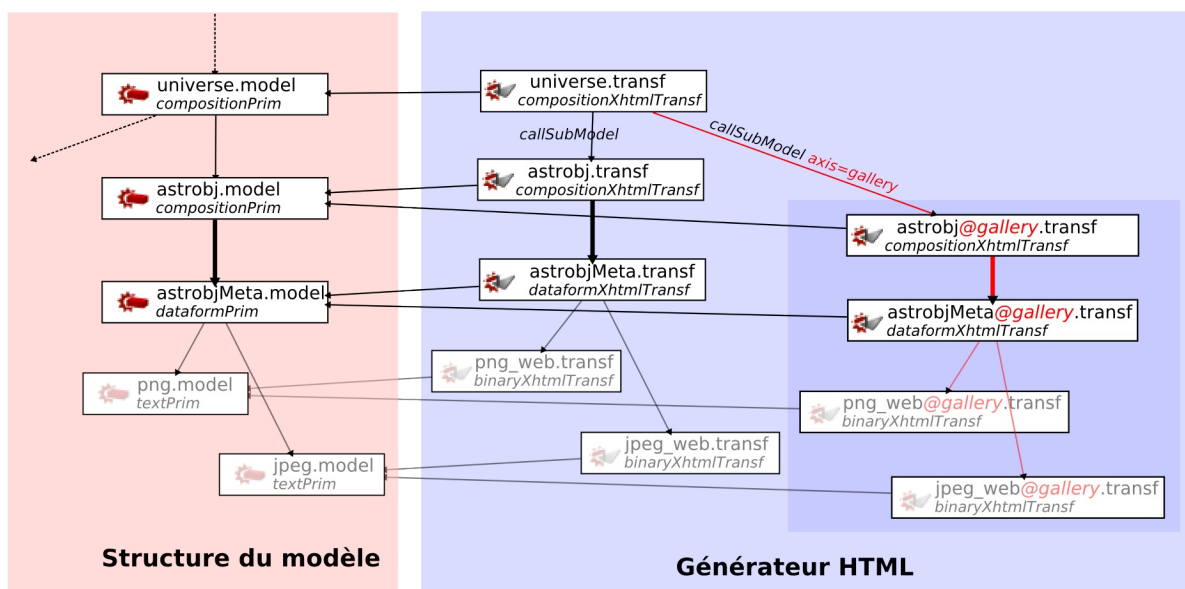
Enregistrez les items en cours et jetez un œil à l'écran de contrôle. Quel est le problème ? Quelle étape manque ?



Attention

Pour passer d'un transformeur à un autre en restant sur le même axis, il faut que celui-ci soit spécifié en deux endroits

- Dans l'entête du transformeur : `<sm:axis code="...">`
- Dans l'appel aux autres transformeurs à partir de celui-ci, par exemple :
`<sm:callCompositionMeta axis="gallery">`



Les axis pour les ressources binaires

Utiliser les axis sur les ressources binaires permet de spécifier leurs propriétés de publication dans un cas d'usage, par exemple changer la taille d'une image dans la version "galeries de miniatures".



Nous allons utiliser cette possibilité pour forcer la taille des images à 200 pixels de large (200 en min et en max). On copie les transformeurs `jpeg_web.transf` et `png_web.transf` du modelet binary pour avoir un point de départ.



on modifie ces 2 transformers.



```
sm:binaryXhtmlTransf info=" "  
  sm:model sc:refUri="image.peg.model (/binaries/jpg)"  
  sm:axis code=" gallery "  
?  
  sm:content  
    sm:WImage widgetClass="galleryobj" crossRefEntries=" "  
      sm:binaryUrl  
        sm:freeTransform code=" " options=" "  
        ?  
        sm:imageTransform  
          ?  
          sm:scale ratio=" "  
          sm:sizeBounds minWidthInPx="200" maxWidthInPx="200" minHeightInPx=" " maxHeightInPx=" "  
          ?  
          sm:outputJPEG  
          ?  
        ?  
      ?  
    ?  
  ?
```

Écran 129



Conseil

On pourrait aussi appeler les transformeurs des images sans axis une fois arrivés dans **astrobjMeta@gallery.transf** mais dans ce cas, nous n'aurions pas l'image à la taille personnalisée que nous souhaitons.




Déclarez les nouveaux transformers créés.

Résultat

Créez un contenu de test avec 3 planètes et autant de photos. Si vous avez réussi, les 3 photos devraient apparaître sur la page d'introduction.



Mettre à jour le fichier transflist

Comme toujours, pensez à rajouter vos nouveaux transformers dans le fichier  transflist de votre modèle (comme vous pouvez le voir sur la fenêtre ci-dessous), c'est valable aussi pour les transformeurs avec axis, **chacun doit être ajouté**.



```
sm:transformerList info="" sc:refUri=""
*
sm:transformerList info="" sc:refUri=""
*
sm:transformer sc:refUri="" universe.transf"
*
sm:transformer sc:refUri="" astrobj.transf"
*
sm:transformer sc:refUri="" astrobjMeta.transf"
*
sm:transformer sc:refUri="" part.transf"
*
sm:transformerList info="" sc:refUri=""
*
sm:transformer sc:refUri="" sTitle_web.transf (/base/sTitle)"
*
sm:transformer sc:refUri="" sTxt_web.transf (/base/sTxt)"
*
sm:transformer sc:refUri="" jpeg_web.transf (/binaries/jpeg)"
*
sm:transformer sc:refUri="" png_web.transf (/binaries/png)"
*
sm:transformer sc:refUri="" jpeg_web@inline.transf (/binaries/jpeg)"
*
sm:transformer sc:refUri="" png_web@inline.transf (/binaries/png)"
*
sm:transformerList info="" sc:refUri=""
*
sm:transformer sc:refUri="" astrobj@gallery.transf"
*
sm:transformer sc:refUri="" astrobjMeta@gallery.transf"
*
sm:transformer sc:refUri="" jpeg_web@gallery.transf"
*
sm:transformer sc:refUri="" png_web@gallery.transf"
*
```

Écran 130

Vous avez tout compris ? transformez votre galerie en galerie cliquable...

> "cf Pour aller plus loin dans la galerie", page Erreur : source de la référence non trouvée.

6 Stylage HTML

Apprendre les CSS

Il existe une multitude de sites internet traitant le sujet, voici ceux que je peux vous recommander :

- [Leçons CSS sur HTML.net](#) (très complet, convient parfaitement aux débutants ambitieux, le sujet est traité de A à Z)
- [Tutoriel de création d'un style pour une page simple](#) (si vous ne savez pas par où commencer après avoir suivi les leçons, c'est un bon exemple, la méthodologie pour faire des skins HTML pour modèles Scenari est similaire)
- [Aide HTML / CSS avancé](#) (astuces sur des sujets précis et pointus)
- [Ressources supplémentaires](#) (programmes conseillés, graphisme...)



6.1. Repartir de zéro



Créer un nouveau style complet



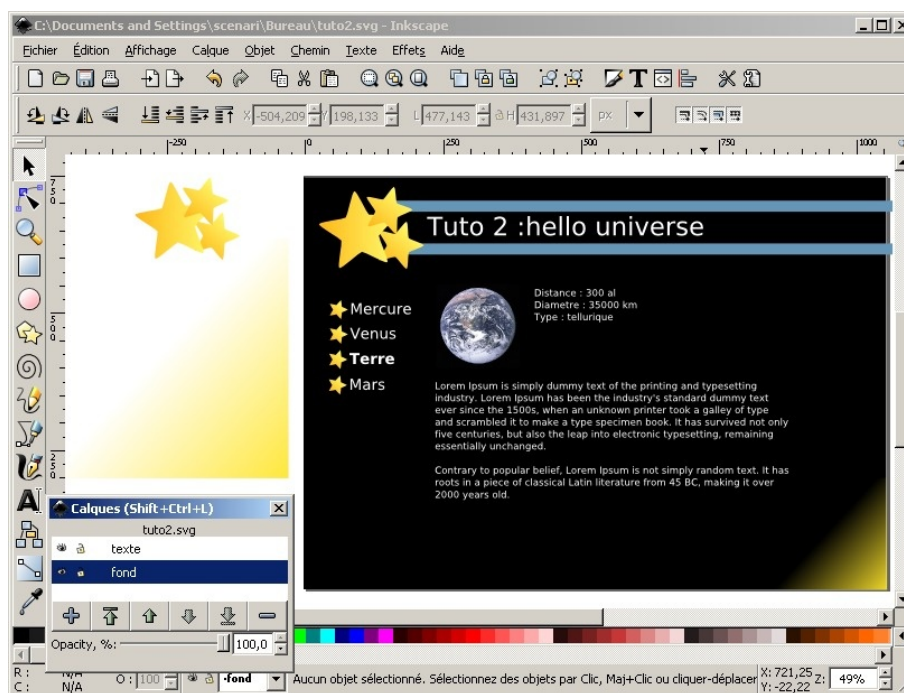
Pré-requis

nécessite d'avoir suivi le tutoriel en entier et/ou une formation dédiée.



Première maquette

Un graphiste crée une "maquette", une image d'une page du site dans un programme **vectériel**. Si plusieurs pages ont des présentations différentes, il crée alors plusieurs maquettes. L'aspect graphique d'une maquette doit être validé rapidement (surtout si vous faites ce travail pour un client). Il est toujours possible de changer un logo à la fin, mais passer une colonne de l'autre côté de la page, ou transformer par exemple un style "à taille fixe 800x600" en un style "ajusté à la taille de la fenêtre du navigateur" demande beaucoup de travail supplémentaire.



Écran 131



Simplification / Validation

Vous demandez au graphiste des simplifications en fonction de ce que vous pouvez ou ne pouvez pas faire en CSS. Certains navigateurs comme internet explorer ont des défauts qui transforment des "idées évidentes" en "casse tête d'intégration" sous certaines circonstances (objets alignés par rapport au bas de la page, hauteurs des colonnes homogène). Ne soyez pas trop ambitieux. L'aspect technique doit être validé par vous même, vous devez avoir en tête les différents blocs qui composeront l'image finale. Faites valider aussi les changements graphiques de ces simplifications.



Faire le template



Créez ou modifiez les template correspondants, ainsi que les fichiers CSS avec les différents blocs, sans y mettre les images pour l'instant. Retournez à l'étape précédente si vous n'arrivez pas à positionner les éléments correctement. Testez la compatibilité avec différents navigateurs, rajoutez des bordures "de debug" pour voir les limites des blocs.



Export d'une maquette finale

Vous spécifiez au graphiste la résolution des images "maquette finale" qu'il fournit, des versions différentes pour bien faire apparaître toute partie d'image que vous allez récupérer (par exemple, sans le texte, sans les "puces du menu", tout ce qui recouvre l'espace)... Précisez lui le format de l'image et sa résolution (png ou jpeg très haute qualité, vous allez devoir le recompresser après).



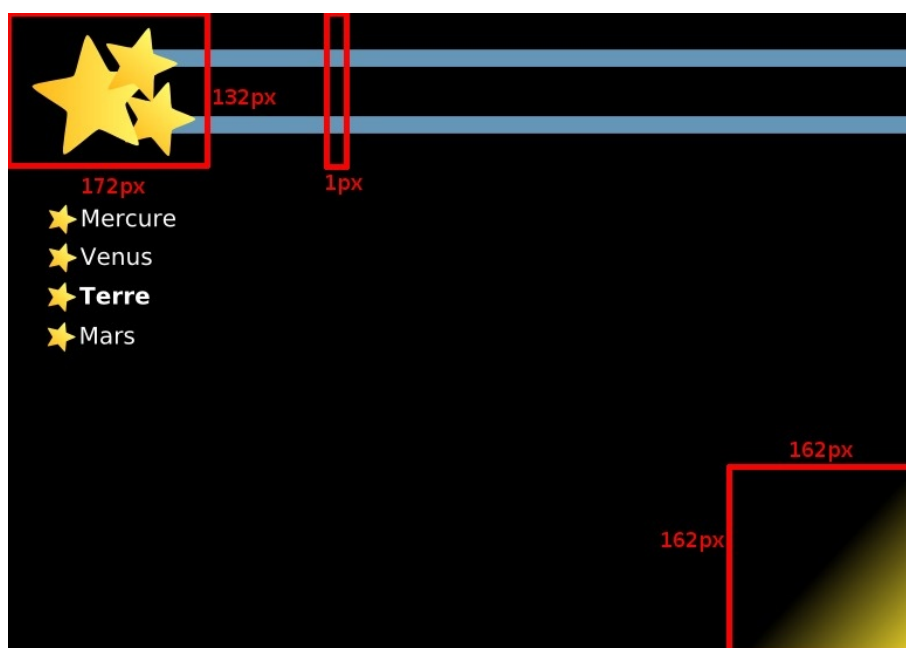
Écran 132



Découpage du fond



Tracez sur papier les blocs à taille fixe de la page, et isolez les avec un outil de traitement d'image (par exemple The Gimp, outil "crop"), tout en notant leurs coordonnées sur papier, pour que vous puissiez reproduire le même découpage ou découper d'autres blocs de même hauteur / largeur. Pour les blocs de taille variable, votre image de fond doit être répétable, ou à défaut avoir une taille maximale et apparaître correctement même tronquée. Souvent, on découpe une ligne de 1 pixel de large qui va être étirée. Ou alors, le plus simple reste juste les blocs avec une couleur de fond sans image. Enregistrez chaque "image de bloc".



Écran 133



Découpage des objets

Il vous reste à découper et intégrer les puces et autres images "flottantes" du contenu.



Intégration dans le CSS

Pour chaque div devant accueillir une image découpée, à vous de régler au moins les propriétés en lien avec l'image de fond, par exemple :

```
<background-position: top left;>  
<background-repeat: no-repeat;>  
<background-image: url(../img/topleft.png);>
```

Pour les noms de fichiers images, repérez vous bien dans l'arborescence des fichiers du site web HTML généré, chaque ".." représente un répertoire parent.

Pour les blocs fixes, je vous recommande de donner **des id de classe** et des noms de fichiers en fonction de la position, ce qui rend votre layout réutilisable. Par exemple, si vous appelez "**topleft.png**" le bloc des étoiles, c'est peut être moins évocateur que "**etoile.png**" mais vous pouvez facilement remplacer l'image par une autre complètement différente si vous devez décliner votre modèle pour traiter un autre sujet.

Exemple avec le bloc "topleft" : les étoiles en haut à gauche :

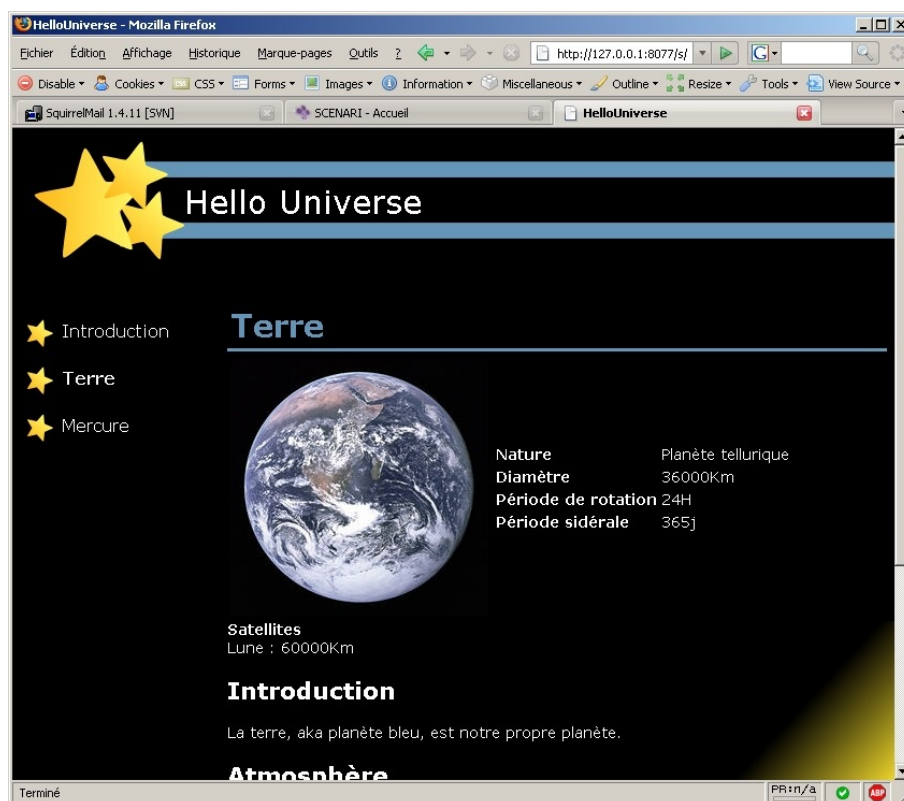
```
#topleft {  
    background-position: top left;  
    background-image:url(img/top-left.png);  
    background-repeat:no-repeat;  
    height:100%;  
    width:172px;  
}
```

Écran 134



Résultat

Vous devriez obtenir quelque chose qui ressemble à l'écran ci dessous, tout à fait conforme à ce que nous avons prévu :



Écran 135

6.2. Modification d'un style existant



Pré-requis

Cette partie a été volontairement simplifiée pour vous permettre de modifier un style sans avoir lu tout le reste du guide. Vous devez cependant :

- avoir récupéré le modèle avec un système de partage de fichiers (par exemple, sur un repository subversion) : [Procédure sur le wiki](#)
- avoir créé l'atelier dans SCENARibuilder en choisissant d'utiliser le répertoire de source que vous venez de télécharger : [Procédure sur le wiki](#)




Repérer les items concernés



Vous partez de l'arbre à gauche, l'explorateur des items, ouvrez les dossiers, et trouvez le fichier  **.wpsdef qui identifie votre modèle et qui va servir de point de départ pour explorer le modèle.**



Pour chacun des items listés, repérez les générateurs  (icône de casquette de chantier). Essayez de vous repérer par rapport au "<name>" de chacun des items ou par rapport au contenu pour "trouver le bon", ou demandez au modélisateur de vous expliquer lequel vous devez utiliser pour arriver à faire vos changements.



Conseil

N'hésitez pas à poser vos questions sur les forums de scenari-platform.org.



Retrouvez les **.doss associés au générateur (directement ou via ses template ou via l'uiFrame)**

Se repérer dans les fichiers CSS

Dans le premier exemple HelloWorld, j'avais cité 2 extensions firefox pouvant être utile au "styleur". J'en donne ici des exemples plus concrets pour montrer comment les utiliser. Il n'est pas nécessaire d'utiliser les 2 simultanément, mais en maîtriser un est un gain de temps considérable pour tout "styleur" sérieux.

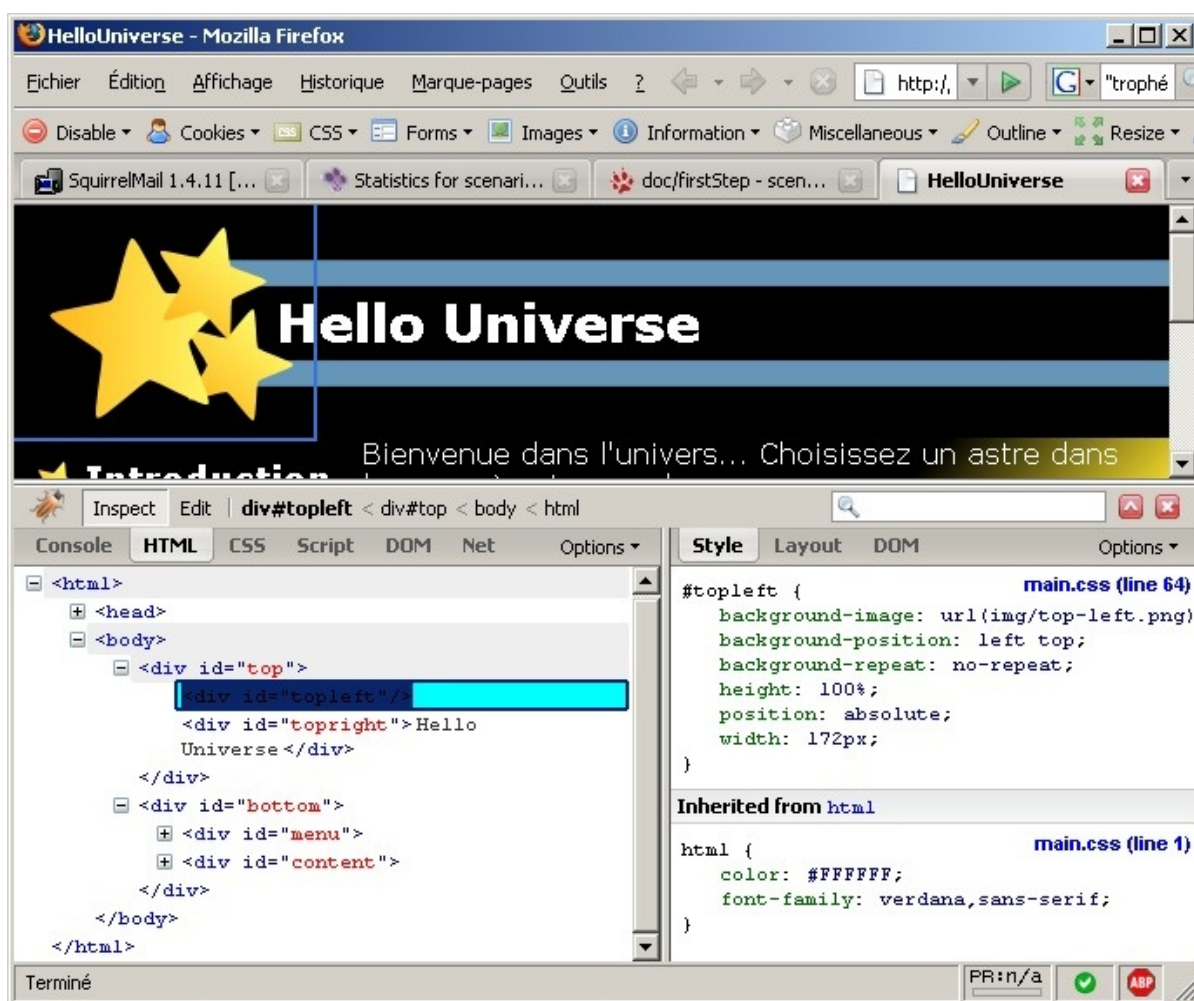
FireBug

Source : <http://www.getfirebug.com/>



Firebug est spécialisé dans l'exploration des arbres d'éléments HTML (le DOM) et dans l'affichage des différentes relations d'héritages de classes CSS. Il permet d'avoir la liste de toutes les déclarations CSS qui vont avoir de l'influence sur un objet précis, et de les modifier directement pour en tester l'effet. En fonctionnalité supplémentaire, il permet par exemple de "debugguer" les javascript en affichant la ligne d'erreur et des informations relative au script.

Pour l'ouvrir, pressez [Ctrl] + [Alt] + [K], puis, choisissez le bouton **inspect** en haut à gauche du nouveau cadre "firebug" qui vient d'apparaître et enfin cliquez sur un élément de la page.



Écran 136

En résultat, on peut voir l'arborescence des éléments HTML jusqu'à notre div "#topleft" qui représente l'image dans le coin en haut à gauche de l'écran, et l'ensemble des styles appliqués à cet objet dans la partie de droite.

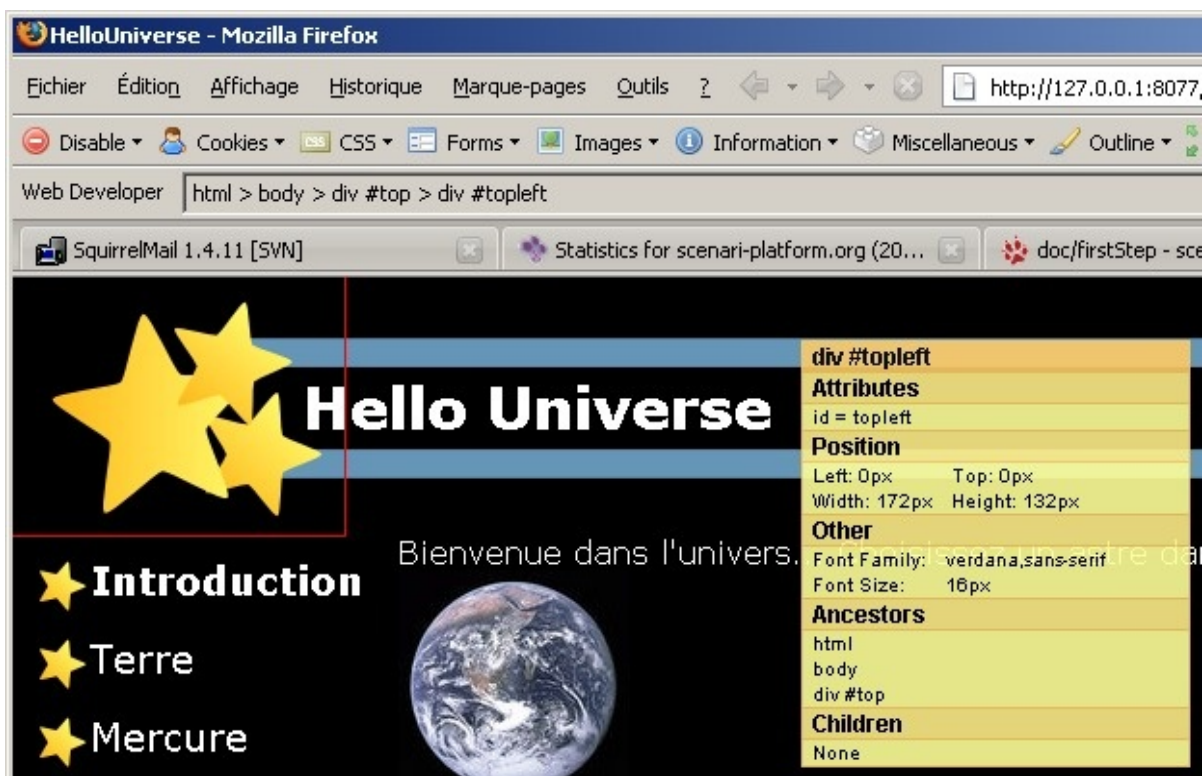
WebDeveloper

Source : <https://addons.mozilla.org/fr/firefox/addon/60>



WebDeveloper est une extension multi-fonctions pour tout ce qui concerne le développement web. Il offre des informations assez basique sur le nom ou l'id d'un bloc sélectionné à l'écran, c'est le plus simple à utiliser. En bonus, des fonctionnalités comme par exemple passer des pages local au validator W3C en un seul clic.

Une fois installé, pressez [Ctrl] + [Alt] + [F] puis cliquez sur un bloc pour déclencher l'affichage d'une fenêtre d'information.




Écran 137

En résultat, on a tout en haut en premier titre de la boîte jaune, l'ID ou la classe de la div, puis les coordonnées ou la liste des noms de styles des conteneurs de cette div.


Test du nouveau style



Pour exploiter le nouveau style vous devez :

1. Lancer un environnement de test :
 - créez un nouvel item  **wsptest**
 - et déposez-y par drag & drop le wspdef.



- ▶ Puis, lancez l'opération de compilation à l'intérieur de ce wsptest.
- 2. Créer un pack de modèle installable dans SCENARichain :
 - ▶ créez un nouvel item  **packMake**
 - ▶ déposez-y par drag & drop le wspdef.
 - ▶ Lancez l'opération de compilation à l'intérieur de ce wsptest,
 - ▶ Révélez ou téléchargez le résultat, installable dans SCENARichain.

7 Stylage OpenDocument Text

Pour trouver de l'aide

Tutoriels

Si vous avez déjà une petite expérience d'utilisation d'un traitement de texte ou d'OpenOffice.org, vous pouvez devenir expert en la maîtrise des styles en suivant ce tutoriel Framasoft : [Créer ses propres styles avec OpenOffice.org](#). Ce tutoriel est très complet, et regroupe des informations sur le stylage des caractères, stylage des titres, de la numérotation... Gardez juste à l'esprit que dans le cadre d'une utilisation Scenari, vous devez partir des styles existant à l'intérieur de l'odStyle et non créer les vôtres.

2ème source de tutoriels : le wiki Scenari. Des parties sont spécifiques à la réalisation de certaines tâches avancées de réalisation de générateur OD.



Forums

Pour développer vos compétences de créations d'odStyle, 2 choses sont à prendre en compte :

La maîtrise d'OpenOffice.org : la grande richesse des fonctionnalités d'OpenOffice.org ne riment pas forcément avec parfaite intuitivité d'utilisation. Fort heureusement, OpenOffice.org est aussi l'un des outils les mieux documentés et avec une communauté d'utilisateurs très active à travers 2 forums :

- Forum OpenOffice.org / writer Anglais : <http://www.oooforum.org/forum/viewforum.phtml?f=2>
- Forum OpenOffice.org / writer Français : <http://www.forum-openoffice.org/forum/forum3.html>



L'intégration avec Scenari peut encore faire l'objet de quelques faiblesses. Si vous obtenez le résultat souhaité dans OpenOffice et dans l'odStyle, mais que lors de la génération votre style n'est pas exploité comme il se doit ou que la génération est en erreur, à défaut d'erreur de manipulation, vous pouvez être victime soit d'un bug, soit d'une fonctionnalité manquante, soit d'une limite de Scenari. Dans ce cas, n'allez pas ennuyer les gens de la communauté OpenOffice mais adressez vous à l'équipe Scenari à travers le forum "modélisateurs".

- Forum [scenari-platform.org / modélisateur](http://scenari-platform.org/modelisateur) : <http://scenari-platform.org/forum/viewforum.php?f=17>

Quelque soit le forum, essayez de chercher avant si votre problème a déjà été évoqué, et si vous devez poster, essayez de formuler votre question avec précision, contexte et détails.



Reconstruire les styles

Vous l'aviez deviné ? Commencez par reconstruire le fichier de styles à partir du fichier **huOd.generator** et remplacer l'ancien par le nouveau.

Mise en page

Accéder aux styles des pages

Vous pouvez choisir d'éditer un style de page en sélectionnant dans la barre de stylage le mode "page" :



Pour chaque pagemaster, un style correspondant a été intégré au fichier de styles (dans notre cas, "first" pour la page de garde et "content" pour les autres).

Propriétés

Nous pouvons intervenir de différentes manières sur le style des pages :

- Changer le format de page (A3, enveloppes, paysage...)
- Modifier les marges
- Choisir une couleur ou image de fond
- Activer / masquer les headers / footer
- ...



Filigrane et image de fond sur toute la page

Normalement, si vous choisissez un fond de type "photo grand format" pour votre page, il ne va pas recouvrir toute la surface de la page, il ne recouvrira que le contenu et pas les marges. Pour y arriver, il existe une astuce :

- Choisissez une taille très petite pour les marges



- Ajoutez une bordure de page d'un pixel de couleur blanche
- Modifiez le "spacing to content" (espace avec le contenu) pour cette bordure

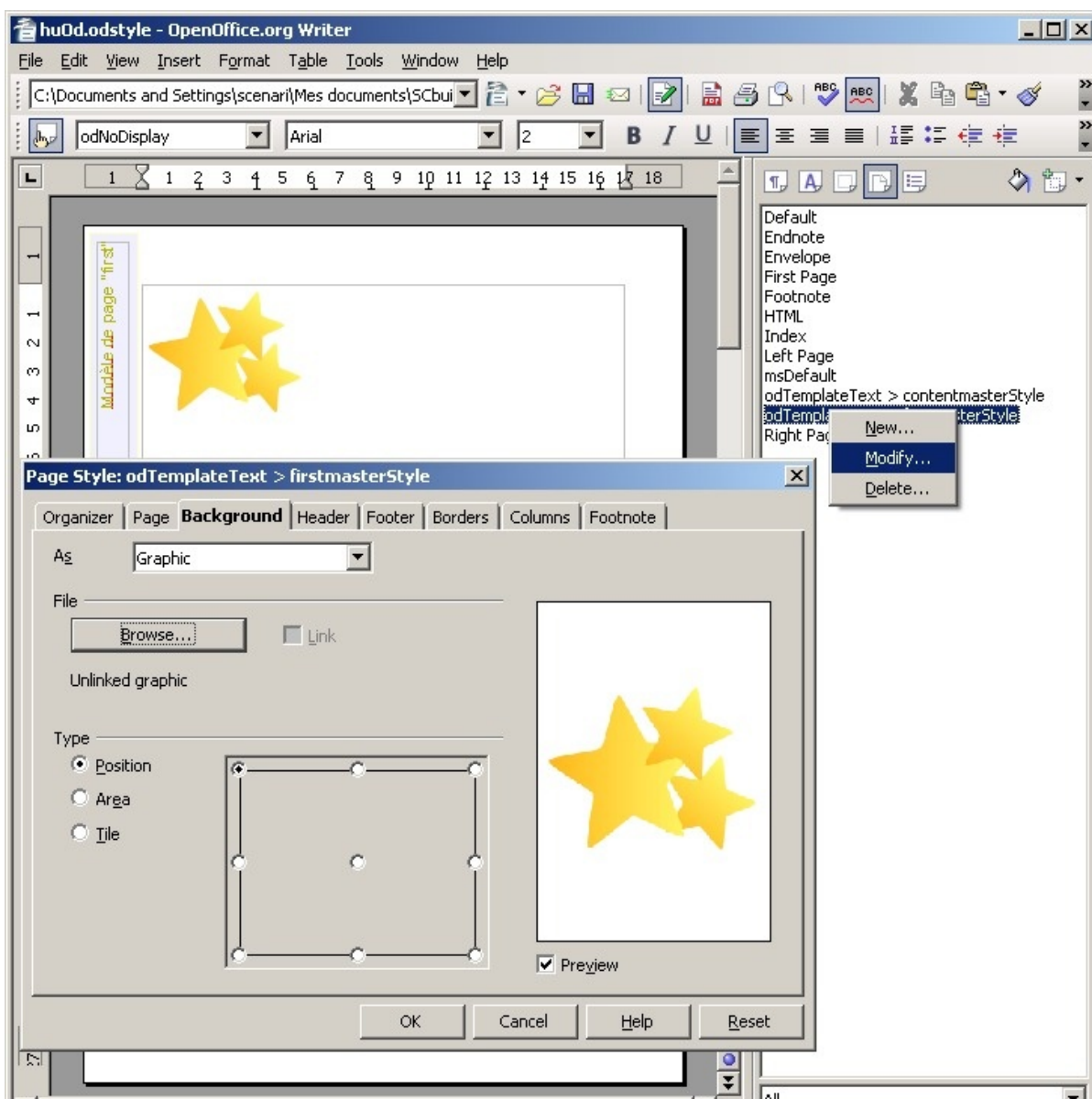
Cette "marge intérieure" sera recouverte par votre image de fond.

OpenOffice.org ne permet pas directement de changer votre image de fond avec un aspect "transparent", augmentez sa luminosité avec the-gimp ou créez un bloc blanc avec alpha à 50% sous inkscape pour obtenir cette effet, avant de sélectionner cette image en tant qu'image de fond dans OpenOffice.org.

Ces astuces proviennent de : <http://homepage.ntlworld.com/pesala/Home/html/watermarks.html>

Pour notre exemple

Pour notre exemple nous choisissons simplement une image positionnée en haut à gauche sur la page de garde :



Écran 138

Sommaire et titres

Où trouver les styles ?

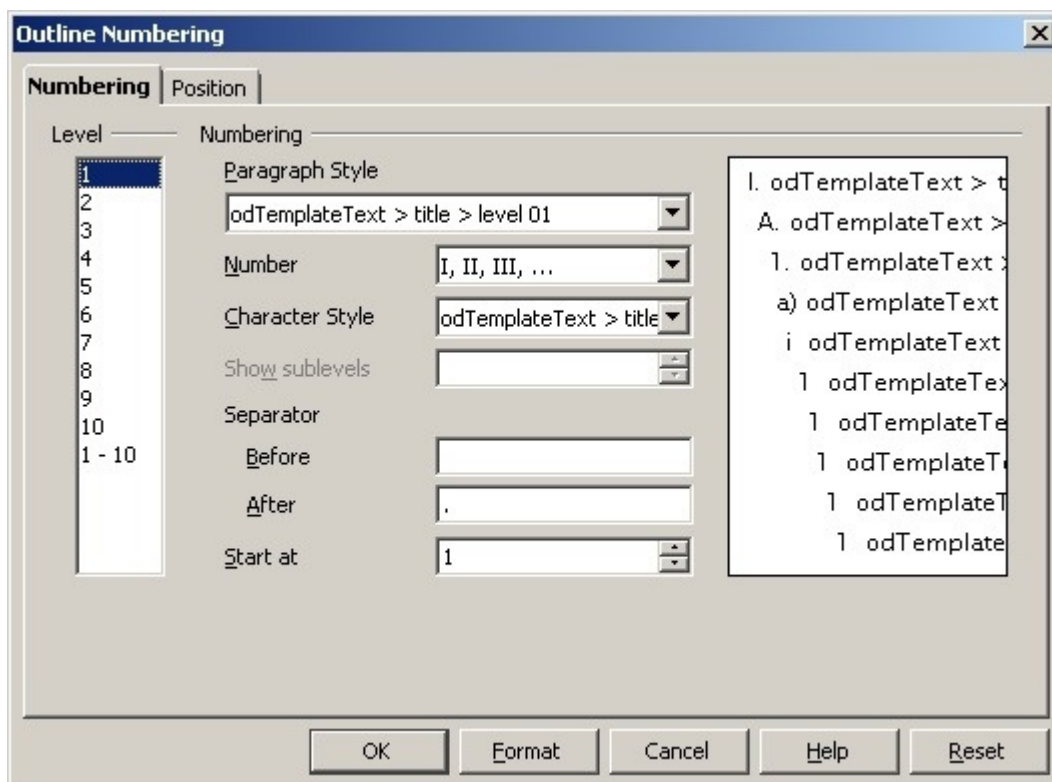
- Style des titres de parties : comme montré dans le premier exemple helloworld, cherchez l'un des premiers blocs jaunes Template "text" (dans notre cas la troisième page du document)
- Style des titres dans le sommaire : à chercher dans le modèle de page du pagemaster correspondant (dans notre cas, la deuxième page)
- Propriétés spécifiques aux titres de parties : menu **Tools -> Outline Numbering**
- Propriétés spécifiques au sommaire : cliquez sur une ligne de contenu à l'intérieur du sommaire, puis simulez la création d'un nouveau sommaire : **insert -> Indexes and tables -> Indexes and tables**



Numérotation et marge des titres dans le contenu

Pour supprimer la numérotation, placez vous dans l'écran **Outline Numbering** :

- Mettez le type **Number** à **None**
- Supprimez les séparateurs "**Before**" et "**After**"... Attention, par défaut "**Before**" contient un espace vide qu'il faut aussi supprimer si vous voulez que votre texte soit bien aligné.

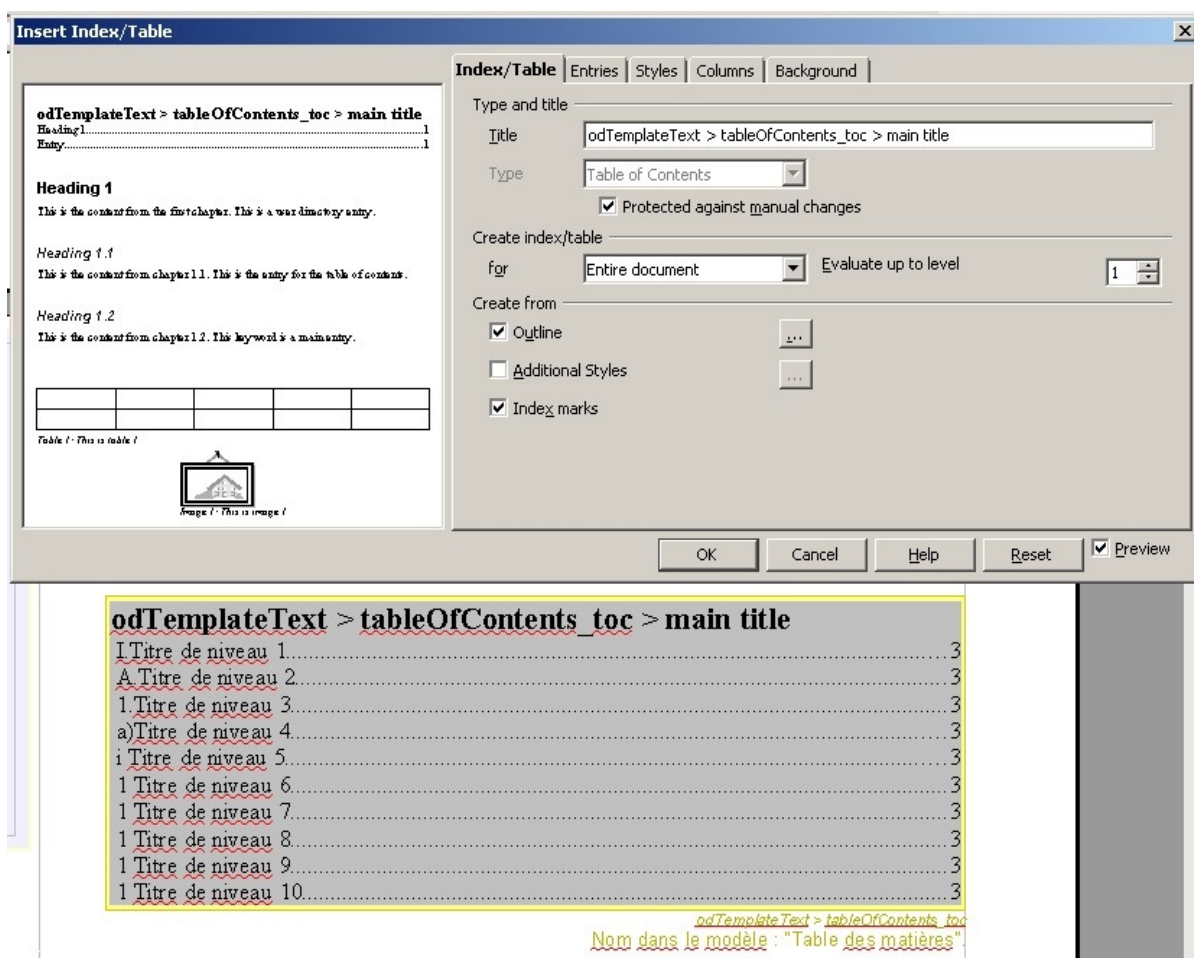


Écran 139

Le 2ème onglet de la boîte de dialogue "position" nous permettrait de régler la marge des titres : même si le style de la police est "aligné à gauche et sans marge", les titres ont une marge supplémentaire définie par cette onglet position. Si nous souhaitons mettre tous les titres alignés à gauche au même niveau, nous devrions passer par cet onglet.

La table des matières

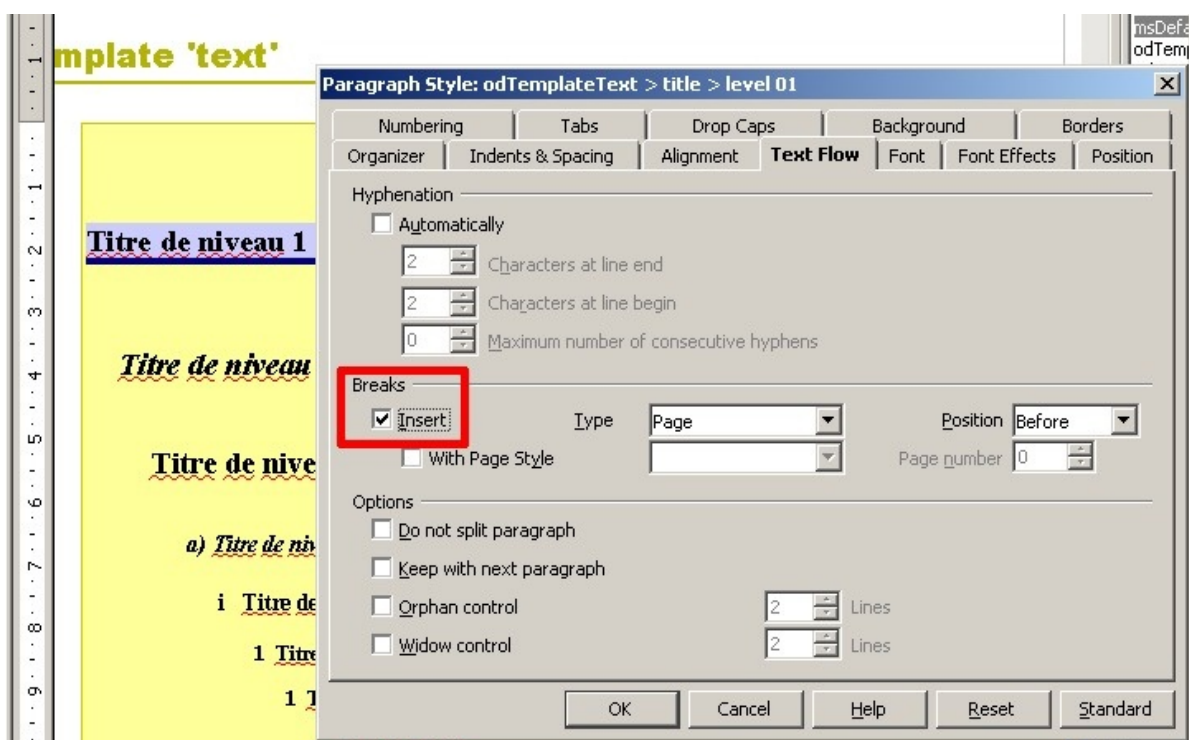
Nous avons déjà réglé précédemment l'option **Evaluate up to level** à 1 pour afficher uniquement le nom des planètes, ce qui est la seule personnalisation au niveau de cet écran dont nous avons besoin pour l'instant. On peut aussi modifier le style de caractère de ces titres.



Écran 140

Saut de page avant le titre de niveau 1

Pour faire un document sous la forme "1 page par planète", nous modifions le style du titre de niveau 1 pour le faire précéder d'un saut de page :



Écran 141

8 Stylage Interface Auteur



sm:authoring

Les paramètres se situent dans la zone **sm:authoring** de vos **.model**.

En effet jusqu'à présent nous avons essentiellement travaillé sur la partie `<sm:structure>` pour définir le schéma documentaire de notre modèle.

Gestion des items

Icônes

L'une des premières personnalisation à apporter sur son modèle est de changer les icônes des items (les pages bleu et rouge par défaut). Le format doit obligatoirement être le suivant : PNG 18x15, 32 bits (avec canal alpha). Les petits symboles **item en erreur** y seront automatiquement superposés s'il y a lieu.

L'icône doit être mise par glisser déposer dans `<sm:icon>` d'un **.model** de type **composition** ou **dataForm**.



Noms

Il est possible de paramétrer le nom par défaut d'un type d'item de Composition dans `<itemNameCreator>`

Zone d'édition : `sm:formEditor`

Si vous avez essayé SCENARIdiscovery, vous avez vu que les titres bénéficiaient d'une mise en forme spéciale. La procédure pour arriver à ce type d'effet est détaillée ici :

- [Wiki : stylage de la zone d'édition Scenari](#)

La meilleure méthode d'apprentissage du stylage auteur reste l'inspiration d'un modèle existant. Récupérez les sources d'un modèle dont l'éditeur vous semble bien stylé, et lancez vous, testez, essayez.

Quelques informations cependant : les champs de stylage comprennent le CSS, les **key** pour le stylage d'éditeur fonctionnent comme les **axis** pour les transformers.



Au niveau de `sm:structure`


1. Choisissez bien vos **name** qu'ils soient explicite pour vos auteurs.
2. Ajoutez des aides contextuelles `<sm:help>`.
3. Choisissez la famille **sub-level** pour exploiter la vue **Plan**.
4. Choisissez bien quelle partie peut être internalisée ou non.

Ces éléments ont une influence forte sur l'ergonomie de l'application et sur l'adhésion future de vos auteurs.

9 Constructeur d'application



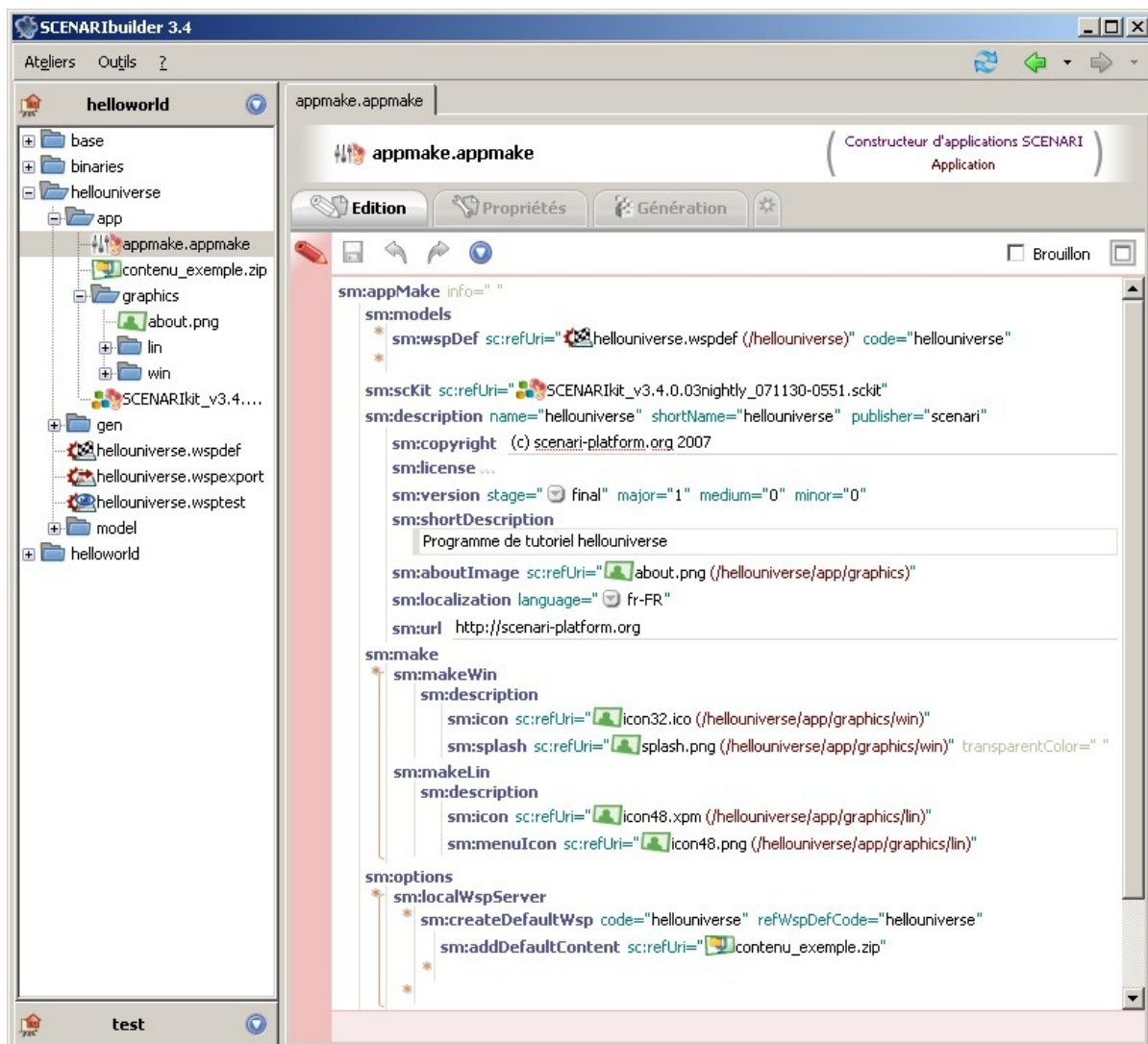
appMake

Pour transformer vos modèles en applications autonome, on utilise un fichier  **.appMake** : un constructeur d'application. Son usage est décrit sur la page du wiki [appMake](#).

pour hellouniverse

Pour **hellouniverse**, nous allons faire une application pour Linux et Windows. Nous devons :

- Créer les icônes à l'aide de **The Gimp** et de parties de notre maquette **Inkscape** ;
- Définir 2 entrées dans `<sm:make>` pour générer les applications pour 2 systèmes d'exploitation : Linux et Windows ;
- Créer un atelier par défaut avec du contenu exemple.



Écran 142



Pour une diffusion sur SCENARichain ou client/server

Utilisez un item de type **packMake** pour diffuser votre application en local sur SCENARichain, en mode distant avec SCENARIservier/client.



10 Bilan

Bravo !

Vous voilà avec une première application Scenari.

Vous souhaitez continuer ?

L'interface de SCENARibuilder les sources de Chaîne Éditoriales, les forums, ces tutoriels devraient vous permettre d'approfondir en **autodidacte** les différents concepts et les différentes possibilités de Scenari. Nous vous conseillons comme pour tout apprentissage de langage informatique de débiter sur **un projet peu ambitieux mais pratique**. Cela vous permettra de vous confronter aux réalités de la modélisation, sans forcément rentrer d'emblée dans les fonctionnalités avancées de Scenari.

N'hésitez pas à travailler **en binôme** avec un modélisateur expérimenté, afin de poursuivre votre apprentissage tout le long de votre projet.

IV Compléments

Cette section regroupe des procédures et concepts qui peuvent vous être déjà connus de part votre expérience avec Scenari. Pour cette raison, ces éléments n'ont pas été introduits dans le guide directement, mais sont à votre disposition comme rappels pour les uns ou comme explication complémentaire pour les autres.

1 Interface générale

Code

Un code est une chaîne de caractères alphanumériques non accentués, sans espace ni caractères spéciaux. Exemple : "legalBlock" mais pas "Bloc réglementaire".

Atelier

Environnement de travail, rassemblant l'ensemble des données. constituant les documents.

Deux ateliers sont hermétiques entre eux, ils ne peuvent partager aucune donnée.

Le contenu de l'atelier s'affiche à gauche de l'écran, dans le volet d'exploration.

Structuration de l'atelier

L'atelier contient les fichiers (ou items) créés par l'auteur.

Un atelier contient au moins un espace.

Espace

Un espace permet de classer des items, de les hiérarchiser. Il est possible d'inclure des espaces dans d'autres espaces, cela forme alors ce qu'on appelle une arborescence.

Le contenu d'un espace s'affiche à gauche de l'écran, dans le volet d'exploration

Structuration d'un espace

Un espace contient des items et peut contenir des espaces.



Truc & astuce

Les espaces se comportent comme des dossiers dans un système de fichiers.

Il est possible de créer une arborescence d'espaces, pour rassembler les items de même thématique, de même catégorie (images, définitions...) et de même type (jpg, png...).



item

un item est un objet de la chaîne éditoriale respectant un certain **schéma de données**, pouvant être **réutilisé** dans plusieurs documents.

Il est matérialisé dans le volet d'exploration.

Les items sont classés dans les espaces et apparaissent dans le volet d'exploration.

Un item peut :

- contenir l'information créée et structurée par l'auteur,
- identifier une ressource créée par des éditeurs tiers (Open Office par exemple).
- référencer d'autres items,
- permettre de créer un support.

Il est matérialisé par un fichier ou un dossier (contenant un ou plusieurs fichiers).



item racine

Item de départ d'un document, d'une modélisation, plus généralement d'une arborescence complète d'items.



Afficher les propriétés de l'atelier

Statut

Les propriétés de l'atelier vous permettent de transmettre à vos collègues, au forum ou au support les informations concernant votre atelier : modèle documentaire utilisé, version installée de ce modèle, adresse physique de stockage sur votre disque dur, etc.

Cliquer sur le bouton d'action , un menu contextuel s'ouvre.

Choisir *Propriétés...*



Définition

L'onglet définition vous permet de changer de modèle documentaire ou d'activer une extension.



Actualiser l'atelier

L'actualisation de l'atelier permet de mettre à jour l'affichage de cet atelier en accord avec son stockage.

Plusieurs méthodes peuvent être utilisées :

- ▶ utiliser la touche [F5]
- ▶ utiliser le bouton de rafraichissement (deux flèches imbriquées bleues), situé en haut à droite de l'interface)
- ▶ utiliser le menu contextuel de la barre d'atelier en passant par le bouton d'actions > *Actualiser*



2 Éditeur



interne (contenu)

Un contenu interne est une structure logique comprise dans la structure de son item parent.



externe (item)

Un item externe est un contenu faisant l'objet d'un item autonome, apparaissant dans le système de fichiers.



Contenu internalisé

désigne un contenu saisi et enregistré dans la grille de saisie d'un item structurant un document.



Contenu externalisé

désigne un contenu faisant l'objet d'un item distinct de l'item utilisé pour structurer un document. L'item correspondant apparaît dans le volet d'exploration et peut être réutilisé dans le document en cours ou dans un autre document.

Conclusion

Merci d'avoir lu ces tutoriels

Parmi tous les visiteurs du site scenari-platform.org (quelques centaines par jour), seulement un pourcentage infime auront le **courage**, le **temps**, l'**ambition**, les **moyens**, d'aller aussi loin dans l'apprentissage de la création de modèles documentaires. C'est un grand gage de motivation que nous apprécions beaucoup.



Conseil

Comme indiqué précédemment, les **forums** et **listes de diffusion** vous sont toujours ouverts si vous avez des questions, si vous voulez vous présenter, ou parler de vos projets avec Scenari. Nous essaierons dans les moyens qui nous sont donnés de répondre au mieux à vos questions, dans l'objectif que leurs traitements **bénéficient à tous** (et ne soient pas du pur debug de votre code spécifique). Ainsi nous espérons que vous aussi vous trouverez le temps de répondre plus tard aux questions de nouveaux modélisateurs.

Pour contribuer

Il existe différentes manières parfois assez simple de faire progresser le projet Scenari :

- **Faites connaître** Scenari à d'éventuels futurs utilisateurs ou modélisateurs.
- **Signalez les difficultés** que vous avez rencontrées lors de l'installation ou l'utilisation de Scenari et les bugs éventuels.
- **Testez les versions alpha ou les nightlies** et si vous avez des propositions d'améliorations très simples, faites des suggestions (de préférence avant que la version suivante n'entre en phase beta pendant laquelle les efforts sont concentrés sur la correction des bugs).
- Testez **Scenari sur votre distribution Linux** ou sur votre environnement, créez et devenez mainteneur des packages Scenari pour cette distribution.
- Placez (si vous le pouvez) vos modèles **en licence libre** et faites en profiter les autres.
- **Rédigez des tutoriels ou de la documentation** en fonction de vos expériences.
- Proposez de réaliser des **traductions** de cette documentation, du site scenari-platform.org ou des plaquettes de communication.



Si vous représentez une entreprise :

- Choisissez des membres de la [communauté scenari-platform.org](http://communauté.scenari-platform.org) ou réclamez l'usage de



Scenari à vos prestataires de service si vous devez sous traiter des projets de production de documents structurés.

- **Aidez au financement du développement** des nouvelles fonctionnalités qui vous intéressent en nous contactant via les forums ou en direct, cf nos coordonnées sur le site communautaire scenari-platform.org ou sur le site professionnel : scenari-enterprise.com).
- Suivez une formation professionnelle Scenari⁴ : vous pouvez reprendre l'apprentissage au départ ou demander l'approfondissement de certains thèmes. Ces tutoriels ciblent un profil « auto didacte » et « développeur », alors que les formations ont des pré-requis plus souples et vous permettent d'aborder pas à pas chaque concept d'une manière plus globale sur l'ensemble des applications Scenari.

4 - <http://scenari-enterprise.com/co/academy.html>



| | |
|----------------------------|--|
| Contenu externalisé | désigne un contenu faisant l'objet d'un item distinct de l'item utilisé pour structurer un document. L'item correspondant apparaît dans le volet d'exploration et peut être réutilisé dans le document en cours ou dans un autre document. |
| Espace | Un espace permet de classer des items, de les hiérarchiser. Il est possible d'inclure des espaces dans d'autres espace, cela forme alors ce qu'on appelle une arborescence. |
| externe (item) | Un item externe est un contenu faisant l'objet d'un item autonome, apparaissant dans le système de fichiers. |
| namespace et prefix | Dans les cas courants, vous allez déclarer toujours le même namespace et préfixe pour tous les fichiers d'un atelier builder. L'intérêt principal de ces informations est de permettre à différents modélisateurs de partager des modèles avec le même nom. Ces noms sont utilisés en interne dans les fichiers de contenu que l'auteur va enregistrer par les éditeurs Scenari. Le préfixe doit être court et ne pas commencer par la lettre "s" qui est réservée (s pour scenari). |
| Racine | Une racine d'un modèle documentaire est un .model pointé par un générateur. La racine principale n'a pas de modèle "parent", elle porte la structure du document. |
| Standalone | On dit d'une application qu'elle est standalone lorsqu'elle peut s'exécuter sans avoir besoin qu'une autre application soit lancée. Les SCENARlapp sont des applications standalone. Un modèle diffusé sous forme de wsppack n'est pas standalone parce qu'il nécessite l'utilisation de SCENARlchain ou du couple SCENARlserver/SCENARlclient pour être lancé. SCENARldiscovery est une application standalone. Opale et OpaleSup existent à la fois en version standalone ET en version wsppack. |