

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Florian Zysset  Leeroy Brun |

Gestion de tâche en XML

Table des matières

1. [Introduction 2](#_Toc401497732)
2. [Cahier des charges 3](#_Toc401497733)

[Tâche](#_Toc401497734)

[Personne](#_Toc401497735)

[Lieu](#_Toc401497736)

[Evénement/Condition](#_Toc401497737)

[UI](#_Toc401497738)

1. [Modélisation 7](#_Toc401497739)

[Tasks](#_Toc401497740)

[Resources](#_Toc401497741)

[Locations](#_Toc401497742)

[Events](#_Toc401497743)

[UI](#_Toc401497744)

1. [Document Type Definition 10](#_Toc401497745)

[Tasks](#_Toc401497746)

[Resources](#_Toc401497747)

[Locations](#_Toc401497748)

[Events](#_Toc401497749)

[UI](#_Toc401497750)

1. [Schéma XML 13](#_Toc401497751)

[Tasks](#_Toc401497752)

[Resources](#_Toc401497753)

[Locations](#_Toc401497754)

[Events](#_Toc401497755)

[UI](#_Toc401497756)

1. [Conclusion 16](#_Toc401497757)

[Fonctionnalités intéressantes mais indisponible](#_Toc401497758)

[Point intéressant de notre modélisation](#_Toc401497759)

[En conclusion](#_Toc401497760)

1. [Annexe 16](#_Toc401497761)

# Introduction

Dans le cadre de notre formation à la HEIG-VD, et durant le module “AppliServ”, il nous a été demandé de modéliser en XML un utilitaire de gestion de tâches.

Nous avons ensuite dû y ajouter des feuilles de style xsl afin d’afficher élégament les informations de ce fichier XML.

Ce document décrit ce que nous avons choisi d’afficher, pourquoi et comment nous l’avons fait.

# Cahier des charges

Le but est de permettre l’affichage facile des données contenues dans le fichier XML, ce qui inclus :

* Liste de toutes les tâches
* Liste des tâches pour une personne donnée
* Détails d’une tâche donnée
* Diagramme des tâches ainsi que des relations entre elles

Nous allons décrire maintenant chacun de ces aspects.

## Liste des tâches

La liste des tâches permet d’avoir une vue d’ensemble du projet en cours.

Il est important d’y afficher les informations de chaque tâche de façon visuelle :

* Resource assignée à la tâche
* Progrès : affichage d’une barre de progression
* Emplacement
* Durée
* Statut : affiché avec des couleurs reconnaissables, permet d’avoir rapidement une vue d’ensemble

Il est important de pouvoir accéder rapidement au détail de chaque tâche en cliquant sur le titre de celle-ci.

Nous devons aussi pouvoir afficher les tâches assignées à une personne en cliquant sur son nom.

Ceci étant le point d’entrée de notre « application », nous devons également avoir des liens vers :

* Le diagramme des tâches
* Le filtre des tâches pour chaque personne

## Liste des tâches pour une personne donnée

Cette page doit permettre de sélectionner une personne et d’en afficher les tâches auxquelles elle est associée.

Cela permet d’avoir rapidement une vue d’ensemble du travail en cours pour une personne sur un projet.

La liste des tâches d’une personne donnée sera présentée de la même manière que la liste des tâches principale.

## Détails d’une tâche donnée

Cette page permet d’afficher les détails d’une tâche donnée.

Les informations qui seront affichées sont les mêmes que pour la liste des tâches, ainsi que les commentaires associés à la tâche.

## Diagramme des tâches ainsi que des relations entre elles

Il s’agit d’une illustration SVG représentant les différentes tâches du projet, ainsi que des relations qu’elles ont entre elles (les events).

Il est important d’afficher sur chaque relation les conditions de l’événement, afin de savoir ce qui lie chacune des tâches.

Il doit être possible de cliquer sur une tâche pour en afficher les détails.

# Implémentation

Cette partie va décrire comment nous allons implémenter chacune des fonctionnalités décrites dans le cahier des charges.

## Liste des tâches

djnfjefjhefhjef

## Liste des tâches pour une personne donnée

Pour visualiser cette vue, ouvrez le fichier « personTasks.html ».

Les fichiers qui concernent cette vue sont les suivants :

* personsFilter.xsl : génère depuis le fichier XML une liste déroulante <select> avec les personnes
* tasksListPerson.xsl : d’après une variable personId, affiche la liste des tâches de cette personne
* personTasks.html : charge les différents éléments et gère les interactions (onchange)

### personsFilter.xsl

Ce fichier génère une liste déroulante de personnes.

<xsl:template match="resources">

<select id="personsFilter">

<option disabled="disabled" selected="selected">Sélectionnez une personne</option>

<xsl:apply-templates />

</select>

</xsl:template>

Nous allons commencer par matcher le tag « resources » qui contient toutes les personnes du projet.

Nous créons donc un select avec un attribut ID (pour pouvoir le récupérer depuis le fichier HTML) avec une option par défaut.

<xsl:template match="person">

<option>

<xsl:attribute name="value"><xsl:value-of select="./@id"/></xsl:attribute>

<xsl:value-of select="./@firstName"/><xsl:text> </xsl:text><xsl:value-of select="./@lastName"/>

</option>

</xsl:template>

Nous matchons ensuite tous les tags « person » qui correspondront à chaque personne du projet. Pour chaque personne, nous générons donc une <option> avec en value l’ID de la personne et comme texte affiché son nom et prénom.

<xsl:template match="tasks"></xsl:template>

<xsl:template match="events"></xsl:template>

<xsl:template match="locations"></xsl:template>

<xsl:template match="ui"></xsl:template>

Pour finir, nous définissons des templates vides pour chacun des éléments que nous ne voulons pas afficher.

### tasksListPerson.xsl

Ce fichier va filtrer les tâches en fonction de la valeur d’une variable « personId ».

<xsl:variable name="personId">res\_1</xsl:variable>

On match ensuite l’élément « tasks » pour lequel on va afficher le tableau des tâches avec le nom de la personne en cours.

<xsl:template match="tasks">

<h1>Taches pour <xsl:value-of select="//resources/person[@id = $personId]/@firstName"/><xsl:text> </xsl:text><xsl:value-of select="//resources/person[@id = $personId]/@lastName"/></h1>

<TABLE class="tasks-list" cellspacing="0" cellpadding="0">

<TR class="heading">

<th class="description">Description</th>

<th class="resource">Resource</th>

<th class="progress">Progrès</th>

<th class="place">Emplacement</th>

<th class="duration">Durée</th>

<th class="status">Status</th>

</TR>

<xsl:apply-templates />

</TABLE>

</xsl:template>

Pour chacune des tâches on vérifie ensuite si la ressource qui y est associée correspond à la variable « personId », et si c’est le cas, on l’affiche. Sinon, il ne se passe rien (donc on ne l’affiche pas).

Il les affichera ensuite de la même manière que tasksList.xsl décris plus haut dans le document.

<xsl:template match="task">

<xsl:if test="./assignedTo/@resource = $personId">

…

</xsl:template>

### personTasks.html

Ce fichier sert à afficher la liste déroulante générée grâce à personsFilter.xsl, puis lors de la sélection d’un élève dans celle-ci, afficher la liste des tâches auxquelles il est assigné.

Nous avons 3 variables globales dans notre script Javascript : xml, personsFilterXsl, tasksListXsl. Elles permettent de charger les différents fichiers XSL et XML une seule fois au chargement de la page et de les garder en mémoire tout le long du fonctionnement du programme.

function loadXML(filename){

if (window.ActiveXObject){

xhttp = new ActiveXObject("Msxml2.XMLHTTP");

} else {

xhttp = new XMLHttpRequest();

}

xhttp.open("GET", filename, false);

try {xhttp.responseType = "msxml-document"} catch(err) {} // Helping IE11

xhttp.send("");

return xhttp.responseXML;

}

La fonction « loadXML » permet de charger un fichier XML/XSL fournis en paramètre et de le retourner.

function getParameterByName(name) {

name = name.replace(/[\[]/, "\\[").replace(/[\]]/, "\\]");

var regex = new RegExp("[\\?&]" + name + "=([^&#]\*)"),

results = regex.exec(location.search);

return results === null ? "" : decodeURIComponent(results[1].replace(/\+/g, " "));

}

La fonction « getParameterByName » permet de récupérer un paramètre fournis dans l’URL. Ceci nous permet dans ce cas de récupérer un éventuel paramètre « ?personId=xxxxx » dans l’URL pour afficher les tâches d’une personne donnée.

// Transforme le XML avec le XSL et le place dans l'élément du dom ayant pour ID "nodeId"

function transformXmlXslIntoNode(xml, xsl, nodeId) {

var containerNode = document.getElementById(nodeId);

containerNode.innerHTML = '';

try {

// code for Chrome, Firefox, Opera, etc.

if (window.XSLTProcessor) {

xsltProcessor = new XSLTProcessor();

xsltProcessor.importStylesheet(xsl);

resultDocument = xsltProcessor.transformToFragment(xml, document);

containerNode.appendChild(resultDocument);

}

// code for IE

else if (window.ActiveXObject) {

ex = xml.transformNode(xsl);

containerNode.innerHTML = ex;

}

} catch(e) {

return e;

}

}

La function “transformXmlXslIntoNode” prend en parameter un fichier XML, un fichier XSL ainsi que l’id d’un element du DOM dans lequel retourner la transformation du XML par le XSL.

// Fonction appelée lorsqu'on veut charger les taches d'une personne

function displayPersonTasks(personId){

var found = false;

// Parcours l'arborescence des enfants du noeud principal pour trouver l'emplacement de la variable

for (var i = 0; i < tasksListXsl.documentElement.childNodes.length && !found; i++) {

var node = tasksListXsl.documentElement.childNodes[i];

// Dès qu'on trouve la variable, on la modifie et on arrête la boucle

if(node.nodeName == 'xsl:variable' && node.getAttribute('name') == 'personId') {

node.childNodes[0].nodeValue = personId;

found = true;

}

}

// Transforme le XML avec le XSL et le place dans le div #listeTaches

transformXmlXslIntoNode(xml, tasksListXsl, 'tasksList');

}

La fonction displayPersonTasks est appelée lors de l’événement « onchange » sur la liste (défini plus bas dans le document). Il prend en paramètre l’ID d’une personne et transformera le fichier XML d’après le fichier XSL « tasksListPerson.xsl ».

Nous devons commencer par rechercher la variable « personId » dans les enfants du nœud racine (son nom est « xsl:variable » et son attribut « name » correspond à « personId »). Une fois qu’on le trouve, on remplace son contenu par l’ID de la personne fournie en paramètre de la fonction et on arrête la recherche.

Il nous suffit ensuite d’appeler transformXmlXslIntoNode pour transformer le XML avec le XSL et placer le résultat dans le div d’id “tasksList”.

function displayPersons(){

xml = loadXML("tasks.xml");

personsFilterXsl = loadXML("personsFilter.xsl");

tasksListXsl = loadXML("tasksListPerson.xsl");

transformXmlXslIntoNode(xml, personsFilterXsl, 'personsFilterWrapper');

// Est-ce qu'on a un personId en paramètre de l'URL ?

var personId = getParameterByName('personId') || null;

// Si oui, on charge déjà sa liste de taches

if(personId) {

displayPersonTasks(personId);

}

// Lorsqu'on sélectionne une valeur dans la liste de personnes, on charge sa liste de taches

document.getElementById('personsFilter').onchange = function() {

displayPersonTasks(document.getElementById('personsFilter').value);

};

}

La function “displayPersons” est appelée au chargement de la page.

Elle commence par charger les fichier XSL/XML dans les variables globales.

Il transforme ensuite le fichier XML avec le fichier XSL pour récupérer la liste déroulante des personnes. Il place ce résultat dans le div ayant pour ID « personsFilterWrapper ».

Il vérifie ensuite si nous avons passé un paramètre dans l’URL pour définir la personne à afficher, si c’est le cas, on appelle directement la fonction « displayPersonTasks » avec l’ID fournis pour afficher les tâches de la personne.

Pour finir, il défini l’événement « onchange » de la liste déroulante. Lorsque celui-ci est appelé, il va appeler « displayPersonTasks » avec la valeur sélectionnée dans la liste (qui correspond à l’id de la personne de qui afficher les tâches).

<body onload="displayPersons()">

<div class="wrapper">

<p><a href="tasksList.xml">Retour à la liste des tâches</a></p>

<div id="personsFilterWrapper"></div>

<div id="tasksList"></div>

</div>

</body>

Ceci est le code du body de la page web avec les différents éléments dans lesquels les informations seront chargées par le Javascript.

## Détails d’une tâche donnée

jnefhhef

## Diagramme des tâches ainsi que des relations entre elles

wkfe