ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Jonathan Wafellman
SIN, Master student

Developer Manual

xtigerTrans.js

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Jonathan Wafellman
SIN, Master student

Table des matières

Jonathan Wafellman
SIN, Master student

1 Introduction

1.1 Description

The library serves to transform XTIGER templates into HTML representation. It uses a HTML document to represent the transformation to apply. This HTML file has to be developed using a micro-format. The javascript code use simple javascripts, HTML DOM and AJAX.

1.2 Structure of the developer manual

In the second section, I define the structure of the library.

In the third section, I explain the functions of transformations.

In the fourth section I define the acquisition functions.

In the fifth section, I define the utility functions

This manual has to be followed with the file 'xtigerTrans.js'. The code has been commented, this manual serves to explain concepts and detail some parts of the code.

2 Structure

2.1 Object

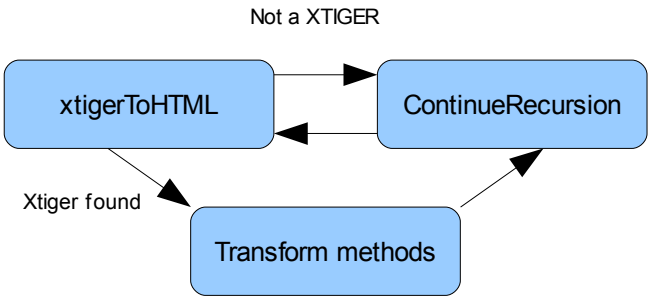
To use the system, you have to create an object that will contain all the structures, types and methods. This object is created using two arguments : the document XTIGER and an URL. The document allow the object to take all the structure inside the document, precisely the components and the unions. The URL serves to take the transformations to apply and the types of the target language.

2.2 Tree walker

The software look for xtigers nodes using a tree walker in deep first search. If it find one, it apply the corresponding transformation. If it doesn't find one, it continue the recursion.

There are two methods, one to recognize the type of node we have, xtigerToHTML(currentNode, newNode) and one that is in charge to continue the recursion, continueRecursion(currentNode, newNode). The seconde method test if we have to continue the recursion.

The function of transformation call directly the function continueRecursion where it is needed.



3 Transformation functions

The generation of the HTML representation depends on the structures we have took from the file defined by the user. In this structure we look for the place where we insert the values asked and where we loop to insert iterative structures. The values are represented by # before their name and the loop are identified using the className of a node.

The simple values are replaced using regular expressions. The values that represent structures are replace using HTML DOM.

In case of loop, the structure under the node that has a className equals to 'loop' are repeated and assign to this node.

Each type's structure is considered like a component and putted in the componentStructs variable of the xtigerTrans object. It is easier to insert them because we don't have to check if we are confronted to a component type, or another type.

In the precedent section, we have seen that the transformation methods call themselves the function to continue the recursion. It is done each time it insert a structure.

4 Acquire structures and types

4.1 Acquire component's structures

Each component can be reused. The system has to know their names and their structures. Names are used like types and they represent the identifier for the component. The structure corresponds to the list of elements to insert when it is called in a use or a bag element.

To find these components, we use the document received during the building of the object. The

system look for each component and extract the needed informations.

4.2 Acquire unions

Unions represent list of types that can be called using only the name of unions. The system take the list of types and translate each union in the corresponding types list. Each element using types will be translated in the same manner during the transformation.

4.3 Acquire transformation structures

This transformations are in a separated file accessed using JAVAX request. From this file, the system takes the body node and then for each child, it verifies if it is a transformation structure using the name of the class. It takes each structure and put it in the object.

In this file, there are types of the target language and special structures that have to be defined each. By default it creates a structure that is simply the name of the type encapsulated by a span node.

5 Utility functions

5.1 UnionToTypes(types)

This function serves to translate each name of union present in a type list into a list of simple types. If unions contain themselves union names in the list of types, it is translated directly during the acquisition. So, the system that translate union name into types list receive and not in these added.

5.2 ExtractInner(node, keep)

The problem using with XML node is that we can't take the string value of what contains the node. This method solve the problem, appending the XML node to a XHTML node and taking the value of innerHTML of this node.

The keep value, that is a boolean, allow user to choose if he wants the value innerHTML of the HTML node, parent of the XML node, or only the innerHTML of the XML Node. It uses regular expression to delete the string value of the XML node.

5.3 replaceNode(oldNode, newNode)

This function corresponds to the function replaceChild(). I've defined it for more clarity but can be easily replaced.

5.4 assignChild(oldNode, newNode)

This function take the children of the oldNode and assign them to the newNode. It works better than using innerHTML attribute. It allows the newNode to keep other children it would have before.

5.5 findLoop(node, loopNodes)

This function serves to find where we have to loop in a transformation's structure. The node corresponds to the structure and loopNodes is the array of node that loops we have found.

5.6 callBack(currentNode, newNode)

This empty function serves to avoid errors if the user doesn't define a callBack function himself.

6 Conclusion

This library translate easily xtiger document in a xhtml representation. Many improvements can be added. The most important is rendering it cross-browser. The need of the user can become more complex. More insertion could be asked or new functionalities during modification of the document.