Arthur Assima

Nordine El Ammari M1 Génie Logiciel

Janvier 2020

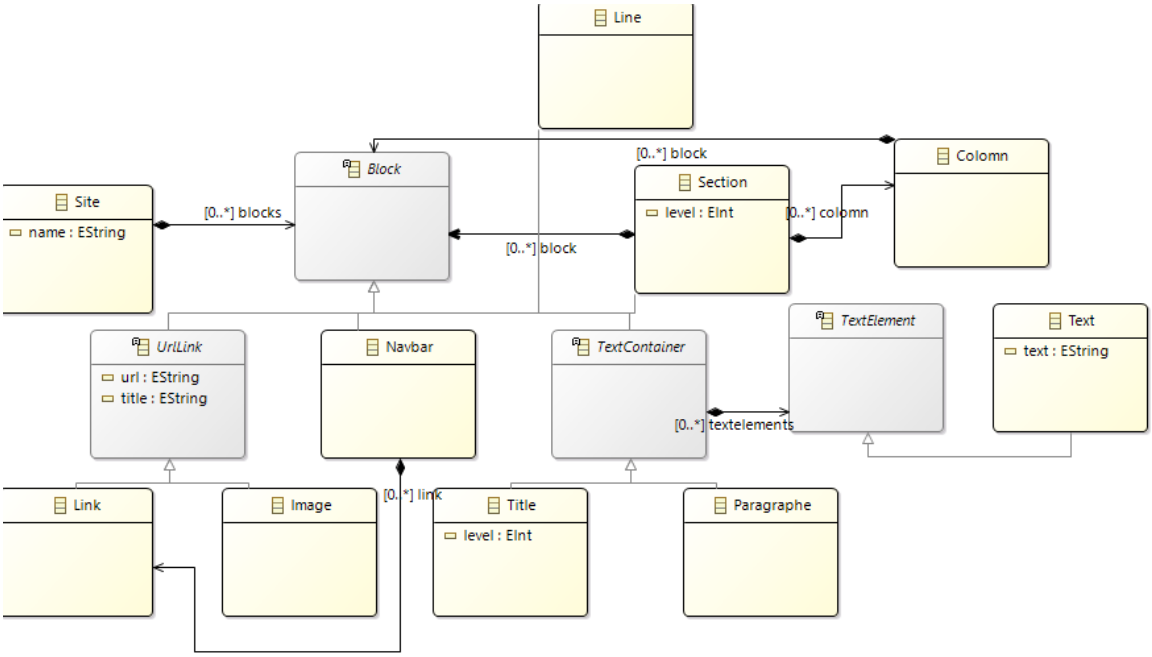
Projet IDM : Documentation Technique

Sommaire

Présentation du projet :

Le projet que nous allons montrer consiste à transformer des fichiers Markdown en fichiers Bootstrap avec une approche IDM, pour ce faire nous avons agencé notre travail en plusieurs parties. Tout d'abord on génère un fichier conforme au métamodèle Markdown (.mds) à l'aide d'un parser. Ensuite nous le transformons conformément au modèle USD (Unified Site Descriptor) et on le transforme une dernière fois en fichier conforme au métamodèle bootstrap. Nous soumettons ce dernier à un fichier Acceleo permettant d'enfin générer le fichier bootstrap correspondant au fichier Markdown de base.

I. Le métamodèle usd



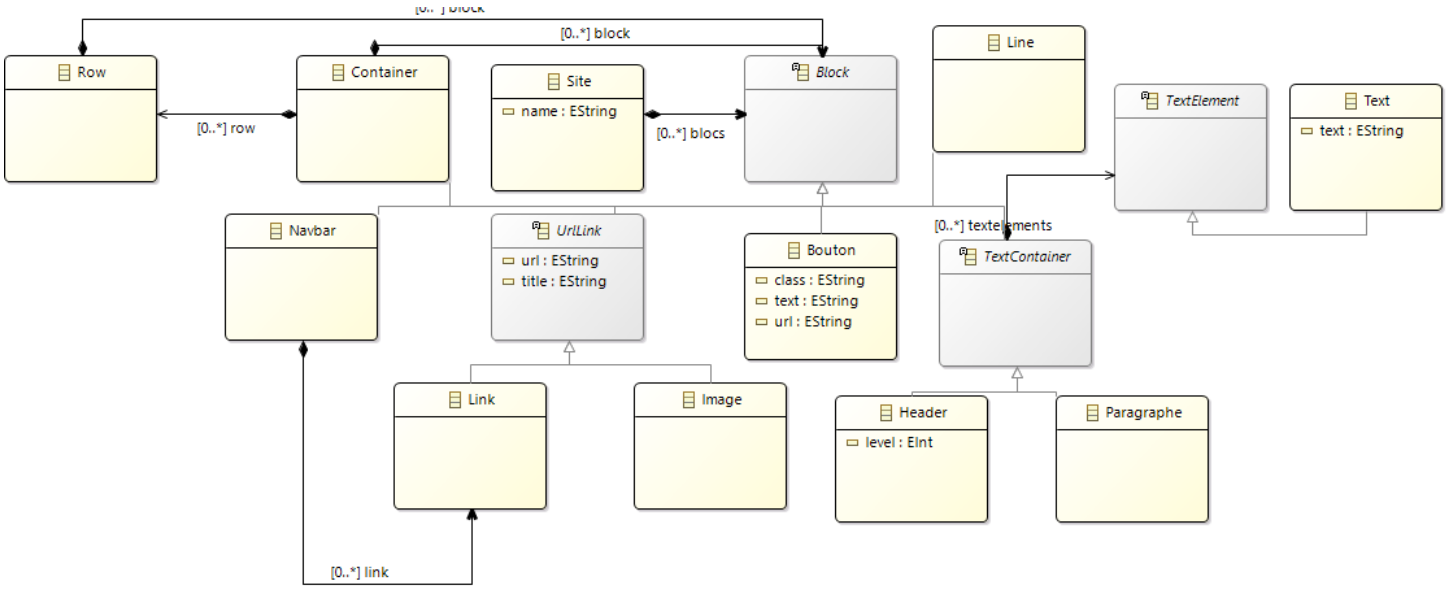
Le métamodèle usd ou métamodèle pivot est un modèle permettant de décrire un site sans trop entrer en détail de manière à permettre une éventuelle transformation vers divers autres modèles de framework tels qu'UML, Foundation ou Bootstrap.

Ce modèle comporte une classe Site composée de Blocks et d'un attribut name. Block est une classe abstraite qui fait office de super-classe pour tous les concepts que l'on pourra trouver dans un site :

- Navbar : une barre de navigation composée de plusieurs lien
- Section : un ensemble de blocks et de Colomns elles même composée de Blocks

- UrlLink : une classe abstraite pouvant représenter un simple lien ou une image
- TextContainer : une classe abstraite pouvant être un Titre ou un Paragraphe qui est composée d'éléments textuels, en l'occurrence un simple Text
- Line : une ligne pour effectuer des séparations.

II. Le métamodèle bootstrap



Le métamodèle bootstrap nous permet de générer un fichier html personnalisé par le framework Bootstrap. Il a des similitudes avec le métamodèle usd mais contient des concepts propres à Bootstrap. Évidemment il ne s'agit que de certains modèles de Bootstrap, pas de la totalité.

Il est également composé d'un Site possédant un nom et composé de Blocks, classe abstraite faisant office de super-classe pour tous les concepts du modèle :

- Bouton : un bouton ayant sa propre classe, son propre lien et son propre texte
- Navbar : une barre de navigation composée de plusieurs liens
- Line : une ligne horizontale permettant d'effectuer une séparation
- Container : une div permettant de stocker d'autres block sous formes de Colonnes
- Header : un entête avec un niveau de taille pouvant varier
- Paragraphe : un paragraphe textuel
- Link : un lien vers un autre site
- Image : une image

III. La transformation md -> usd

Pour passer du métamodèle markdown à usd on attribue à Site le nom du Document markdown et on utilise un mapping block2block() sur chaque Block de Document pour l'ajouter au Block du Site usd.

```
main() {
    var site : Site := object Site{
        md.rootObjects()[Document] -> forOne( doc ) {
          name := doc.name;
          blocks += doc.contents -> map block2block();
        }
        }
}
```

La plupart des transformations de md à usd sont complexes en raison de la présence majoritaire de paragraphe et de title . Le mapping block2block utilise une disjonction de 7 mapping où chaque Paragraphe et chaque Titre est modifié selon les éléments qu'il

contient vers un concept d'usd. Par exemple s'il n'a que des textes il sera transformé en Paragraphe dans usd, s'il n'a qu'un Lien il sera transformé en Lien, etc. (para2link, para2img, title2title, …)

```
mapping markdownmm::Paragraph::para2img(): usdmm::Image
when {self.allSubobjectsOfType(Image) -> size() >0; }{
    self.elements -> forEach (e) {
        url := e.oclAsType(Image).url;
        e.oclAsType(Image).subtext -> forOne (t) { title := t.oclAsType(Text).text };
    }
}
```

On utilise la clause when pour différencier certains contenus des titres et paragraphes markdown.

Les concepts doivent êtres séparés par un saut de ligne pour que le parser ne mélangent pas texte et image(par exemple) dans le même paragraphe du metamodèle markdown.

Malheureusement nous n'avons pas pu transformer une section et ses colonnes car nous n'avons pas trouvé comment transformer uniquement les blocks qui se trouvaient à la suite d'une section jusqu'à ce qu'une autre section commence.

L'unique transformation simple est celle de la ligne horizontale qui n'a pas besoin d'être modifiée.

IV. La transformation usd -> bootstrap

```
main() {
    var site : bootmm::Site := object bootmm::Site{
        usd.rootObjects()[usdmm::Site] -> forEach( s ) {
          name := s.name;
          blocs += s.blocks -> map block2block() ;
        }
    }
}
```

Pour passer du métamodèle usd à bootstrap on utilise le même principe qu'à la transformation précédente, chaque Block du Site usd est transformé en block du site bootstrap par le mapping block2block.

Block2block contient également une disjonction avec des transformations plus simples où on ne fait qu'affecter les attributs car on passe de concept spécifique à concept spécifique (img2img, link2link, para2para).

```
mapping usdmm::Block::block2block() : bootmm::Block
disjuncts usdmm::Title::title2header,usdmm::Link::bouton2bouton,usdmm::Link::link2link,
 usdmm::Image::img2img, usdmm::Paragraphe::para2para, usdmm::Navbar::nav2nav,
 usdmm::Line::line2line {}
```

On ajoute aussi la possibilité de transformer un link en bouton en mettant une clause when avant le mapping (link2button).

V. La génération de code bootstrap

On a utilisé Acceleo pour générer le code boostrap à partir du métamodèle bootstrap. Le template de base est de type Site.

On écrit le squelette d'un fichier html avec les sources bootstrap dans la partie head et on lance le template generateElement dans le body qui parcoure tous les blocs du Site.

Le template generateElement se lance sur le type abstrait Block donc dans cette fonction on crée une condition if pour chaque concept et si la méthode oclIsKindOf est vraie on lance le template du concept correspondant.

VII. Quelques exemples de transformation

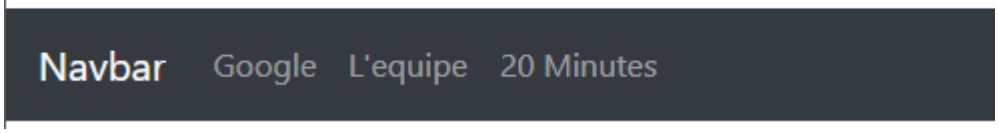
Markdown :

```
Une Navbar :

[[]Navbar](#)
[Google](http://google.com)
[L'equipe](http://lequipe.fr)
[20 Minutes](http://20minutes.fr)
```

Bootstrap :

Une Navbar :

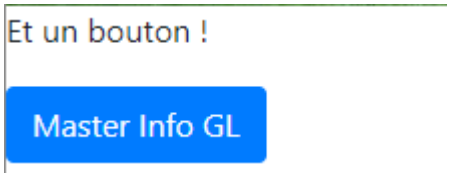


Markdown :

```
Et un bouton !

[[btn btn-primary]Master Info GL](#https://www.fil.univ-lille1.fr/portail/index.php?dipl=MInfo&sem=M1GL&ue=ACCUEIL&label=Pr%C3%A9sentation)
```

Bootstrap :



Conclusion

Nous avons réalisé la chaine de transformation entre markdown et bootstrap avec 7 concepts disponibles en bootstrap :

- Header
- Paragraphe
- Image
- Bouton
- Link
- Navbar
- Line

Ce projet nous a permis d'avoir une perspective différente de ce que nous avons eu à développer jusqu'à présent avec une approche IDM et un langage nouveau qui nous a causé plusieurs difficultés mais qui nous a apporté un autre regard sur la conception des modèles.