An inheritance model for documents in web applications with sydonie

Jean-Marc Lecarpentier, Pierre-Yves Buard, Hervé Le Crosnier, Romain Brixtel

An Inheritance Model for Documents in Web Applications with Sydonie *

Jean-Marc Lecarpentier, Pierre-Yves Buard, Hervé Le Crosnier, Romain Brixtel
GREYC - CNRS UMR 6072
Université de Caen - Basse-Normandie
{firstname.lastname}@unicaen.fr

ABSTRACT

Each web site has to manage documents tailored for its specific needs. When building applications with a specific document model, web developers must make a choice: build from scratch or use existing tools with the need to accomodate the model. We propose an inheritance model for documents, implemented in the Sydonie open source web development framework. It offers a flexible environment to create classes of documents. Sydonie's document model uses entity nodes inspired by the Functional Requirements for Bibliographics Records (FRBR). Document content and metadata are modeled using a set of relations between entity nodes and attribute objects. Classes of documents or attribute types can be defined through a declarative XML file. Our inheritance model provides the possibility to define them at the framework level, application profile level or application level. This demonstration explains the document definition process and inheritance model implemented in the framework and gives several examples of its advantages.

Categories and Subject Descriptors

H.3.2 [**Information Systems**]: Information Storage; H.5.4 [**Information Systems**]: Hypertext/Hypermedia—*Architectures*; I.7.1 [**Document and Text Processing**]: Document and Text Editing—*Document Management*; J.7 [**Computers in Other Systems**]: Publishing

General Terms

Design, Documentation

Keywords

Document Management System, Document Model, Composite Documents, Web Development Framework

*Research work funded by the Conseil Régional de Basse-Normandie with the CPER program, the European Council with the FEDER program, and the TGE-Adonis.

1. INTRODUCTION

Content Management Systems (CMS) are widely used to manage web sites. They provide functionalities to easily create and publish content. Predefined document types and ready-to-use modules allow for customization of the web site or application. CMS usually focus on content creation and publication. With a different approach, Web development frameworks provide tools to help creating web sites or applications. Frameworks provide functionalities to avoid coding common tasks, such as database access, session management, templating, etc. While very useful, these tools do not provide a model for documents and their management. On the other hand, library systems focus on document management, in particular on document metadata. However, they do not manage the content of documents.

In this paper, we present Sydonie, an open source web application framework implemented in PHP. Sydonie takes its roots in the web development community and the library world. Sydonie provides a document model to manage multilingual composite documents. A document inheritance model provides web designers with a flexible development environment.

The remainder of this paper is structured as follows. The next section gives an overview of how some existing CMS or frameworks manage documents and the lessons learnt. Section 3 presents Sydonie's document model, its internal data model and how classes of documents are defined. Section 4 introduces the framework's inheritance model for the creation of custom classes of documents. Examples are given in Section 5. Finally, Section 6 concludes this article.

2. CMS, FRAMEWORKS AND DOCUMENTS

As the name indicates, Content Management Systems deal with *content*. With the Drupal CMS [2], the CCK module [1] allows web site administrators to customize the content of entities by associating field names to a type of content. The administrator builds new content types on top of the core system by adding new fields. In order to create a custom application, a developer will first make his application model fit into the CMS model.

On the other hand, frameworks provide more flexibility by letting the developer design the classes for the application, therefore reflecting the application model. For example, when using Symfony [4], a PHP web development framework, a document type corresponds to a class declaration, with formatted comments to define the relationships to other objects.

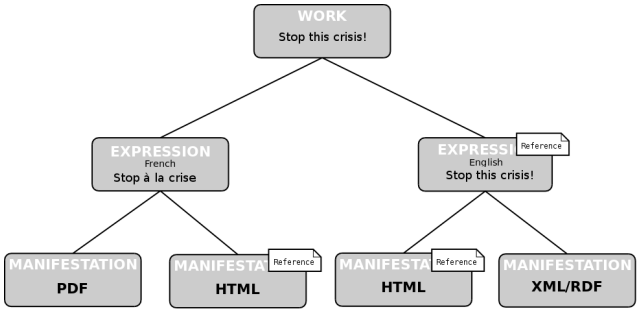

Figure 1: A Manifestation is an embodiment of an Expression of a Work

These systems are well designed to manage content, but are not made to manage *documents*. In the above cases, the system manages content components but does not consider the set of components as a document entity. The approach is to define a document as the rendition of some content [7], where a document is similar to the web page displayed to a user. Using an different approach, a library system considers a document to be a reference card including metadata and a pointer to the document itself through some kind of identifier. These systems manage document metadata but not their content.

We need an hybrid approach to define and manage digital documents. Documents on the web should not be treated as mere content, but as an information container where the information is both content and metadata. The next section presents how the Sydonie framework implements this concept.

3. SYDONIE

Sydonie, a Document Management System for Publishing on the Web[1], is a web development framework [3]. Sydonie is developed within the University of Caen Basse-Normandie, in conjunction with C&F_éditions[2], a publishing partner developing online services based on Sydonie. Sydonie[3] is open source software made available under a GPL license. Implemented in PHP and relying on a mySQL database, Sydonie can run on any basic LAMP server. This section introduces the core concepts of the framework.

3.1 Document model

From the CMS world, Sydonie's document model inherits the approach of using an online editing system to produce a web rendering document (i.e. the HTML version of a document). From the library world, it uses the metadata model and the Functional Requirements for Bibliographic Records (FRBR) [6]. The FRBR conceptual model introduces three groups of entities to capture bibliographic data. FRBR group 1 contains four hierarchical entity levels *Work*, *Expression*, *Manifestation* and *Item* which represent the different aspects of intellectual or artistic works.

Using the guidelines for group 1 entities from the FRBR report, Sydonie defines a document model with the entity

[1]In French: SYstème de gestion de Documents Numériques pour l'Internet et l'Édition

[2]`http://cfeditions.com`

[3]`http://sydonie.net`, under construction

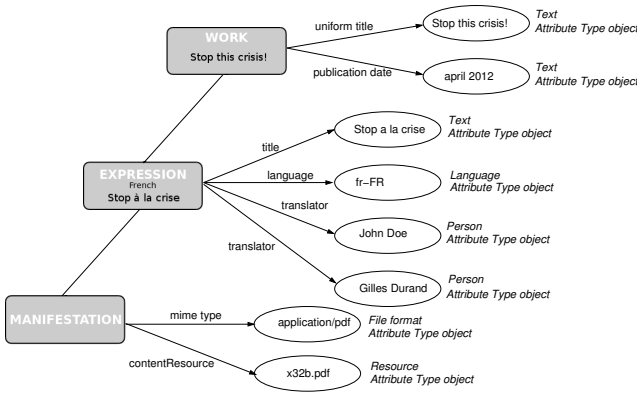

Figure 2: Branch of a document with its attributes

levels Work, Expression and Manifestation to represent intellectual or physical aspects of a document. Sydonie's model considers a document as the complete tree, as shown in figure 1. A document is thus defined by a tree composed of Work, Expression and Manifestation entity nodes. The set of data attached to each node represents a document's data and metadata. Language negotiation and content negotiation are used to determine which Manifestation is to be served to a user. This process is also used within composite documents [5].

3.2 Document attributes

In order to manage any kind of application, a framework must be able to manage different kinds of documents. Using Sydonie's document model, a class of document is the definition of what information each entity level may contain. The kind of information associated to each node may vary depending on the class of document.

To provide a generic way to manage the information attached to a node, Sydonie uses an attribute-value based model. The framework provides a data structure that can adapt to any kind of information to be attached to a node. Each node has a list of attributes where the attribute points to an object that models the attached information. Within the framework, similarly to RDF, attributes are triples (subject, predicate, object) where:

- subject is an instance of a document entity node, i.e. a Work, Expression or Manifestation node;
- predicate is the name of the attribute, i.e. the name of the relation;
- object is the value. It is an object (in the OO sense). Its class models the information it represents. It can also be a list of objects when a predicate represents multiple values.

Figure 2 illustrates this model in the case of an article. The flexibility of the approach resides in the fact that entity nodes are generic objects used by all classes of documents. The `DocumentEntity` class represents a node of the document tree (i.e. either a Work, Expression or Manifestation node). The list of attributes is managed at the framework level and is composed of an array of attributes objects representing the named relations between a document entity and some data. The associated data are instances of objects

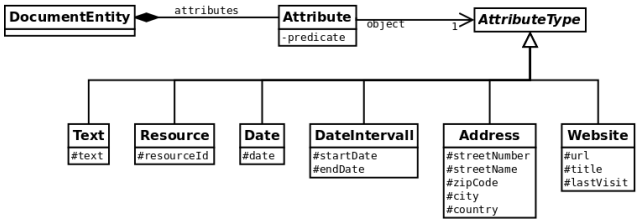

Figure 3: Relations between entity nodes (Work, Expression or Manifestation) and attribute types

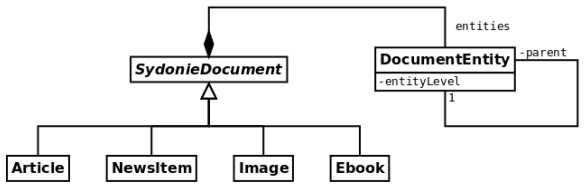

Figure 4: SydonieDocument abstract class and some document types

inheriting from an `AttributeType` abstract class. These children classes can model scalar data, such as text or integer, or more complex structures, such as a price or an address for example. Figure 3 shows the relations between entity nodes and `AttributeType` children classes. The framework provides predefined `AttributeType` classes. New types can be defined at the application level, allowing any document type to naturally "fit in" Sydonie's document model.

To create a class of documents, a developer needs to define what `AttributeType` objects each entity level may accept. The next section shows how to create classes of documents in a declarative manner.

3.3 Classes of documents

As explained in section 3.1, a document is a tree of entity nodes linked to `AttributeType` objects. The framework provides the model and routines for document reification through the abstract class `SydonieDocument`. The `SydonieDocument` class is the base class for all documents in Sydonie. It defines the tree structure of documents, the articulation between document entities (Work, Expression or Manifestation nodes), their attributes and the `AttributeType` objects that contain the document's data and metadata. Figure 4 shows the relation between the `SydonieDocument` and `DocumentEntity` classes. Any class of documents must inherit from `SydonieDocument`. Similarly to `AttributeType` objects, the framework provides some predefined classes of documents, as shown in figure 4.

A class of documents is the definition of what data each entity level nodes may contain. This information, the entity level and the type of data each attribute points to, must be defined in order for the framework to manage the document reification. An XML configuration file defines the needed information in a declarative way, therefore allowing a developer to learn and create a class of documents from existing examples. An example of configuration file is shown in figure 5. To declare a class of document, the configuration file specifies for each possible attribute: the predicate (name of the attribute); the entity level (Work, Expression or Manifestation); the multiplicity (the attribute can appear 0, 1 or more times); the `AttributeType` class that contains the object value and the mandatory information the `AttributeType` will require.

```
<configuration>
   <class>Article</class>
   <extends>SydonieDocument</extends>
   <attribute entityLevel="expression" minOccur="1"
                                       maxOccur="1" >
      <predicate>title</predicate>
      <objectClass>Text</objectClass>
   </attribute>
   <attribute entityLevel="work" minOccur="0"
                                 maxOccur="1000" >
      <predicate>link</predicate>
      <objectClass>Website</objectClass>
      <!-- list mandatory information of
           the AttributeType object.
           Here, only the url property of a Website
           object will be mandatory   -->
      <mandatoryProp><prop>url</prop></mandatoryProp>
   </attribute>
</configuration>
```

Figure 5: Example of configuration file for an `Article` class of documents

Even though the XML file is simple to create, a developer must be able to reuse already defined classes of documents. Sydonie's architecture allows component reuse and customization. Existing classes of documents can be reused and finely tuned using Sydonie's inheritance model, introduced in the next section.

4. INHERITANCE MODEL

Sydonie uses its own inheritance model to determine, at the application level, the information a document may contain. A class of documents can be defined at the framework level or at the application level. Classic object oriented inheritance is not sufficient to manage document definition. For example, the framework's document layer defines a basic `Article` class of documents. Let us suppose that, when creating an application, a developer needs to use article documents with more information (i.e. more attributes). Using classic object oriented inheritance, one could create a `MyArticle` class of documents that inherits from the `Article` class. The `MyArticle` class would then define the changes made to the parent class. The trouble would appear when the developer wishes to reuse some already defined routines for `Article` documents: since `MyArticle` is a different document class, these routines may not work any more. A simple example is listing `Articles`: the framework would then list only the `Article` instances, but not `MyArticle` instances. This behavior would be fine if the application needs a *new* class of documents, but not if it only needs to *alter* an existing class.

In order to allow alteration as shown in the above example, Sydonie provides its own inheritance model. The classic object oriented inheritance model is used to allow the creation of new classes of documents. These classes inherit the properties of their parent class and add or alter properties. Sydonie's inheritance model adds new features to allow fine tuning of existing classes of documents. In our example, the `Article` class inherits the abstract base class `SydonieDocument` introduced in section 3.3. At the framework level, `SydonieDocument` declares common attributes to all classes of documents: `firstPublished` at the Work level, `title` and `language` at the Expression level and `content`-

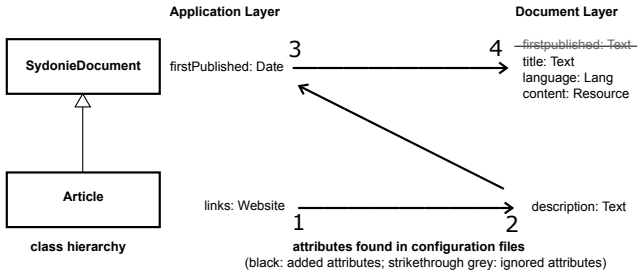

Figure 6: Sydonie's inheritance model

`Resource` at the Manifestation level. This information is specified in `SydonieDocument`'s configuration file. At the framework level, the `Article` class adds the `description` attribute at the Expression level, using classic inheritance (i.e. by specifying that `Article` inherits `SydonieDocument` as illustrated by figure 4). Using the inheritance declaration, the framework compiles these two files to define the `Article` class.

Sydonie's inheritance model, shown in figure 6, named "cascading" inheritance, allows a web designer to enhance the already defined `Article` class. To alter the definition of `Article` in the application layer, the developer creates a configuration file for the `Article` class. The framework will process this file on top of the existing one. For example, the application needs `Article` documents to have the `firstPublished` attribute to be a `Date` attribute type object instead of `Text`, and to have a `links` attribute to add references to web sites (using the `Website` attribute type). The application will only need to specify the changes to `firstPublished` and the addition of a `links` attribute to the `Article` class of documents. Whenever the application uses an `Article` document, the framework will check the presence of a configuration file for each class and at each level, going from child to parent class and from application level to framework level. For each file, it will add the defined attributes that are not defined yet. It is important to note that, if `firstPublished` is specified as `Date` in the `SydonieDocument` configuration file in this application (instead of the `Article` configuration), then the `firstPublished` attribute type would be a `Date` for *all* classes of documents in the application built, as shown in figure 6.

5. CASE STUDY

The proposed model and architecture are implemented in the framework and have been tested with several applications. The Craham[4] is a historical and archaeological research unit at the University of Caen Basse-Normandie. It manages a collection of photographs of archaeological sites, and an application was built with Sydonie to manage the digital versions of the scanned slides[5]. It mostly relies on the framework's image class and uses Sydonie's metadata model and management within images using XMP. This application uses the framework's image class of documents, enhancing its default model with attributes to reflect on the specific data needed (e.g. archeological site, location, etc.).

[4] `http://www.unicaen.fr/crahm/`
[5] `http://craham.info.unicaen.fr` (under development)

C&Féditions created two applications using Sydonie. Polifile[6] is an application to create eBooks online using a WYSIWYG editor. It relies on Sydonie to manage users, and uses Sydonie's document layer for eBooks and images. In the application layer, it interacts with an ePub library to create ePub files. *Mémoire des Catastrophes*[7] is an application to collect witness stories about disasters that occured in France. It uses Sydonie's document layer for images and articles. In the application layer, classes of documents were created to model disasters and witness stories. The next version of the site will use the application profile layer to provide a blog. A third application, companion website of the Net.Lang book[8], is currently under development.

6. CONCLUSIONS

In this paper, we present two main contributions. First, we propose a document model implemented in the Sydonie web development framework. Sydonie uses a tree model based on FRBR and a RDF-like structure to allow documents to contain any kind of information. Document content and metadata are stored at different entity levels to express their level of abstraction. Then, we propose an architecture and development model. The layered architecture and cascading inheritance model is applied for classes of documents, and also for templates, form bindings and interactions, as well as for actions on documents. Sydonie's model focuses on providing web designers with document and application models that they can easily adapt to their specific needs.

7. REFERENCES

[1] Drupal cck module, 2012.
[2] Drupal open source cms, 2012.
[3] Sydonie framework, 2012.
[4] Symfony framework, 2012.
[5] J.-M. Lecarpentier, C. Bazin, and H. Le Crosnier. Multilingual composite document management framework for the internet: an frbr approach. In *Proceedings of DocEng 2010*, page 13, Sept. 2010.
[6] O. Madison. *Functional Requirements for Bibliographic Records*. K. G. Saur, Munich, Germany, 1998.
[7] R. T. Pédauque. *Le document à la lumière du numérique*. C&F éditions, 2006.

[6] `http://polifile.fr`
[7] `http://memoiredescatastrophes.org`
[8] `http://www.net-lang.net`