HANDLING STRUCTURED DATA IN THE ALFRESCO SYSTEM

Goran Sladić, Stevan Gostojić, Branko Milosavljević, Zora Konjović
{sladicg, gostojic, mbranko, ftn_zora}@uns.ac.rs
Faculty of Technical Sciences, University of Novi Sad

Abstract – *This paper describes an extension of the Alfresco system which provides retrieval of data from other relational databases. Retrieved information can be used as metadata for folders, documents and business processes within Alfresco. The proposed solution is compliant with the Alfresco mechanism for defining content models and therefore can provide a data source for different Alfresco installations.*

1. INTRODUCTION

Over the last decade, business systems have been transformed from the manual operation to the computer supported creation and exchange of information. The document management system is an information system that enables the creation, storage, transfer, organisation, retrieval, exchange, manipulation and update of documents [1]. In other words, it is an information system that is used to store and track electronic documents or scans of paper documents. Technologies for the electronic documents management are suitable for use in various business environments [1].

There are five major benefits of the document management system usage [2, 3, 4]:

- **Money Saving.** Quality document management system reduces the paper costs, the costs incurred due to loss of documents, and also its use leads to more efficient use of space that was previously used to store documents. Moreover, time to find documents is reduced, and thus employer productivity can be increased.
- **Provides document security.** This is one of the most important advantages for any enterprise. The electronic document management provides the efficient protection of documents from competitors, thieves and employees that want to use documents for the prohibited purposes.
- **Easy document access.** This has significantly increased efficiency. Document management systems allow multiple users to simultaneously have access to the same document, wherein only one has the ability to change the document. There is also a version control, which provides access to the latest version of a document, but also access to all changes that were made. If needed, users can access to earlier versions.
- **Damage control.** Document management system provides several ways to protect data from loss. One possibility is the backup creation, and the other is export of the repository to another location.
- **Consistency of procedures**. Employees often want to do business in their own way; however, a document management system ensures that employees do theirs jobs according to the protocol.

The basic functionalities of document management systems are [1, 2]:

- support different structural forms of documents and metadata association to documents,
- business processes definition and document lifecycle management,
- indexing and retrieval of documents,
- optical character recognition,
- users collaboration and version control,
- security, and
- integration with other systems.

Today, document management systems often coexist with ERP (Enterprise Resource Planning) systems within the organization. Currently, one of the important functionality that is being developed in the modern document management systems is to bind those systems with ERP systems. One segment of these features is to retrieve data from an ERP system and then use it as metadata for documents in a document management system.

The system described in this paper is an upgrade of the Alfresco system [3], which provides retrieval of data from the ERP's relational database and use of this information as metadata for spaces (folders), documents and business processes within Alfresco.

2. ALFRESCO SYSTEM

The document management systems development has proceeded in the direction of connecting with other components of an enterprise information system. Today, the current view of business information systems development involves the notion of an enterprise content management (ECM) system. ECM is actually term used to describe a collection of content-centric technologies that includes [2]:

- **Document Management (DM)**: Capturing, organising, and sharing binary files. These files are typically produced by office productivity software, but the scope of the files being managed is unlimited.
- **Web Content Management (WCM)**: Managing files and content specifically intended to be delivered to the Web. The key theme of WCM is to reduce the "web developer" bottleneck and empower non-

technical content owners to publish their own content.

- **Digital Asset Management (DAM)**: Managing graphics, video, and audio. This is DM with added functionality specific to the needs of working with rich media such as thumb-nailing, trans-coding, and editing. Like WCM, the intent is to streamline the production process.
- **Records Management (RM)**: Managing content as a legal record. Like DAM, RM starts with DM and adds functionality specific to the world of RM such as retention policies, records plans, and audit trails.
- **Imaging**: This includes capturing, tagging, and routing images of documents from scanners.

The Alfresco system is the web-based ECM system that can be used for content management in small and medium organisations as well as in large, complex geographically distributed organisations [2, 3]. It is licensed under the *GNU General Public License*. Alfresco uses only technologies of the open source software and the open standards. It allows relatively easy customization and expansion of its functionality through the use of XML configuration documents and integration with other applications using the open standards. Moreover, in the case of more complex requirements, the source code of the system can be changed [3].

Alfresco provides the organisation with the necessary services to create, convert, manage and share electronic resources [3, 8]. There is also version control, the ability to search and visualise relationships and dependencies between content. Integrated workflow provides full control over the documents' lifecycle, and managing processes that involve documents [3].

3. ALFRESCO CONTENT MODEL

The first step in the process of designing a custom content management application is creating the content model. The Alfresco content model is used to present all types of content within its repository. The key information presented in Alfresco through this model are [3, 5, 6]:

- fundamental data types and how those data types should be persisted to the database,
- higher order data types (like "content" or "folder") as well as custom content types (like "contract"),
- built-in aspects and custom aspects,
- metadata specific to each content type,
- constraints placed on properties,
- how to index content for searching, and
- relationships between content types

The Alfresco content model is built using a set of following building blocks:

- **Types** – are like types or classes in the object-oriented world. They can be used to model business objects, they have properties, and they can inherit from a parent type.
- **Properties –** represent pieces of metadata associated with a particular type.
- **Property types –** describe the fundamental types of data the repository will use to store properties.
- **Constraints –** can be used to restrict the values that Alfresco will store in a property. They can be defined once and reused across the model.
- **Associations –** are used to define relationships between types. Association provides for defining a hierarchy of folders, containment of a document in a folder, and many other relations between different contents.
- **Aspects –** are mechanism to extend content model with properties and associations by attaching them to content types when and where they are needed. Same as types, the aspects are created using the properties and the associations.

The Alfresco content model can be defined through the XML documents that must be in accordance with the appropriate XML schema defined by Alfresco. One of the drawbacks of this model is that the existing restrictions on properties do not restrict property values on values that are defined in a database table. This paper proposes one implementation of such restrictions, keeping that the previously described method for defining content models in Alfresco system is satisfied.

4. SYSTEM SPECIFICATION

Support to retrieve property values from a relational data model can be achieved by defining the appropriate content model and the Alfresco extension. This extension should provide the handling of structured data retrieved from the database.

4.1 CONTENT MODEL DEFINITION

The content model properties with values that are retrieved from a relational database are created in the same way as any other property of the content model. The only thing that matters in this case is that the type of the property must corresponds to the type of primary key of the table from which data are retrieved, because the property value is the value of the selected row's primary key field.

An example of defining such properties is given in Listing 1. This listing defines a new aspect (*inf:address*) which represents the address in the given content model. Defined aspect consists of three properties: *inf:user, inf:street*, and *inf:city*.

In this example, the properties *inf:user* and *inf:street* retrieve their values from the corresponding database tables. As in those tables the primary key field type is *Integer*, the type of the properties is *d:int.*

```
<model name="inf:infmodel" xmlns="http://www.alfresco.org/model/dictionary/1.0">
  ...
  <namespaces>
       <namespace uri="http://informatika.ftn.uns.ac.rs/alfresco" prefix="inf" />
  </namespaces>

  <aspects>
       <aspect name="inf:address">
         <title>Adress aspect</title>
         <properties>
           <property name="inf:user">
              <type>d:int</type>
              <mandatory>true</mandatory>
           </property>
           <property name="inf:street">
              <type>d:string</type>
              <mandatory>true</mandatory>
           </property>
           <property name="inf:city">
              <type>d:int</type>
              <mandatory>true</mandatory>
           </property>
         </properties>
       </aspect>
  </aspects>
</model>
```

Listing 1. An example of the aspect with the properties that retrieve values from a relational model

4.2. MODULE FOR HANDLING STRUCTURED DATA

Module for handling structured data is developed relying on the Alfresco mechanisms for extending the Alfresco functionality. Therefore, it can be effectively and easily installed in different configurations of the Alfresco system.

For the purposes of the module functionality, the Alfresco database schema has been extended with the table shown in Figure 1. This table provides information for mapping content model properties onto tables from a relational model.

property_models			
id	<pi>	Integer	<M>
qname		Variable characters (255)	<M>
table_name		Variable characters (255)	<M>
id_column_name		Variable characters (255)	<M>
display_name		Variable characters (255)	<M>
order_name		Variable characters (255)	

Figure 1. Table for defining mappings

Value of the column *qname* is the name of the Alfresco data model property. The *table_name* column contains the name of the table from which the values are loaded, while the column *id_column_name* contains the name of the column that represents the primary key in the aforementioned table. The primary key value of the selected column represents the value of the Alfresco content model property. Since the column that represents the primary key very often does not contain any semantics (surrogate key), but only uniquely identifies tuples we introduced the column *dislay_name*.

It contains the name of the column which values are displayed to a user in the appropriate user interface component. The value of this column can be the name of a column, but also it could be a join of two or more columns. The list of displayed values will be sorted by column(s) specified as the value of the *order_name* column.

The class diagram of the given module is presented in Figure 2. The main class in this model is *DBProperties-Generator*. This class is intended for visualisation of the properties whose values are taken from a relational model. Since its functionality relies on the functionality of the existing *TextFieldGenerator* generator, *DBProperties-Generator* inherits this class. In the case when the property's display mode is in the *read* mode, the *UIOutput* class from the standard JSF (Java Server Faces) API [7] is used to display property value. In the case that the application is in the *edit* mode the JSF classes *UISelectItem* and *UISelectItems* are used to display property values. These classes allow a user to select the desired value of the property from the list of values retrieved from a relational database. The class *DBPropertyConverter* performs mapping of the primary key onto the information displayed to a user, i.e. based on the primary key value, it retrieves the data from a database that will be displayed to a user. Complete access to a relational database is carried out by the *DBPropertyService* class, where connection to a relational database is managed through the *ConnectionPool* class. The *DBPropertyModel* class is designed to access the table which defines the mapping. The class *DBProperty-ModelItem* models one item from that table.

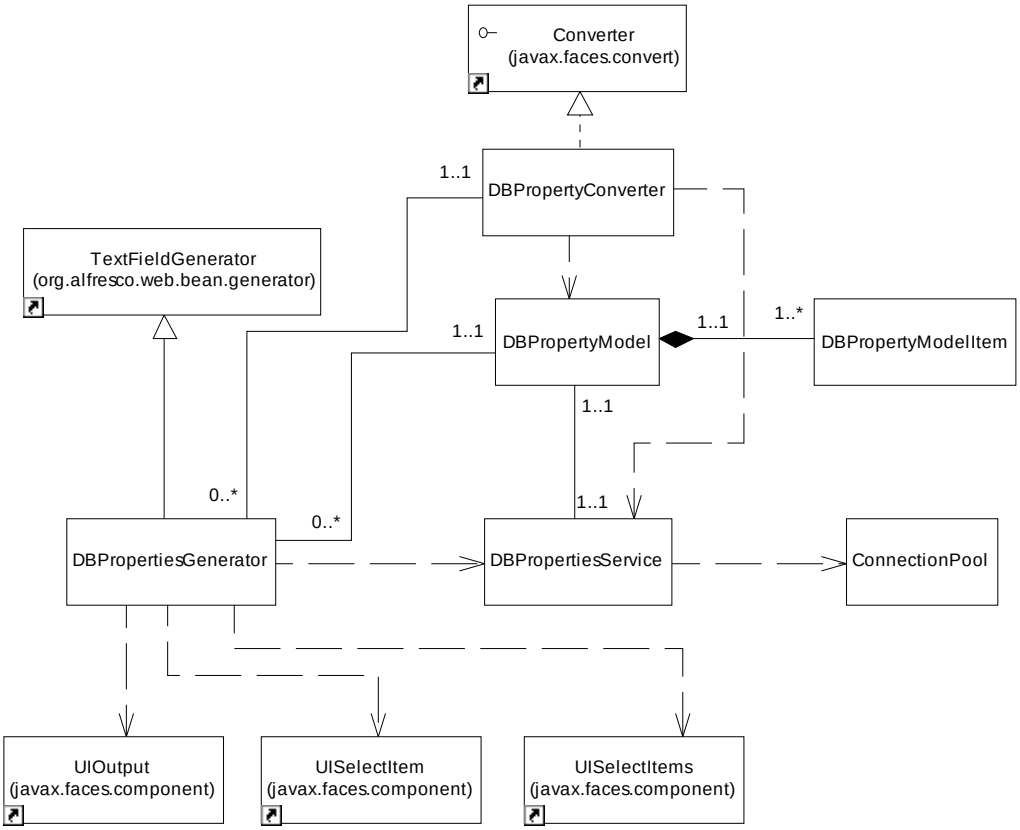

Figure 2. The class diagram of the module for handling structured data

4.3. MODULE INSTALLATION

Installation of this extension includes the registration of the certain module's classes in the Alfresco configuration files [8]. The class *DBPropertiesService* is registered as the Spring bean [9] (see Listing 2), while the classes *DBPropertiesGenerator* and *DBPropertyConverter* are registered as the JSF components [10] (see Listing 3).

```
<beans>
 <bean name="propertyService"
       class="rs.ac.uns.ftn.informatika.
          alfresco.DBPropertiesService"/>
</beans>
```

Listing 2. Spring part registration

```
<faces-config>
  <managed-bean>
    <managed-bean-name>DBPropertiesGenerator</managed-bean-name>
    <managed-bean-class>
      rs.ac.uns.ftn.informatika.alfresco.DBPropertiesGenerator
    </managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
      <managed-property>
        <property-name>propertyService</property-name>
        <value>#{propertyService}</value>
      </managed-property>
  </managed-bean>
  <converter>
    <converter-id>org.alfresco.faces.DBPropertyConverter</converter-id>
    <converter-class>
      rs.ac.uns.ftn.informatika.alfresco.DBPropertyConverter
    </converter-class>
  </converter>
</faces-config>
```

Listing 3. JSF part registration

5. A CASE STUDY

As a case study we will demonstrate the use of the proposed solution on the content model presented in Listing 1. For this content model the table *property_models* is populated with two rows given in Table 1. The property *inf:user* is mapped onto the table *users*. Value for this property is the *id* column, while the display name is concatenation of the *first_name* and *last_name* columns. List of values is sorted by the

last_name, and then by the *first_name* column. The property *inf:city* is mapped onto the table *cities*. The property value is the value of the selected row *id* column. Display name is the value of the *name* column. Also, list of values is sorted by this column.

qname	table_name	id_column_name	display_name	order_name
inf:user	users	id	CONCAT(first_name, ' ', last_name)	last_name, first_name
inf:city	cities	id	name	name

Table 1. Mapping for the given content model

Visualisation of the presented content model, when used as document properties, in the *read* mode is presented in Figure 3, while the same properties in the *edit* mode are presented in Figure 4.

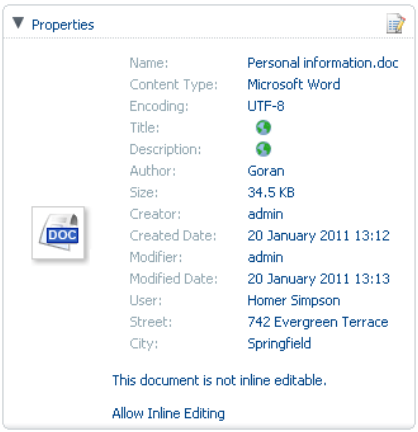

Figure 3. Visualisation in the read mode

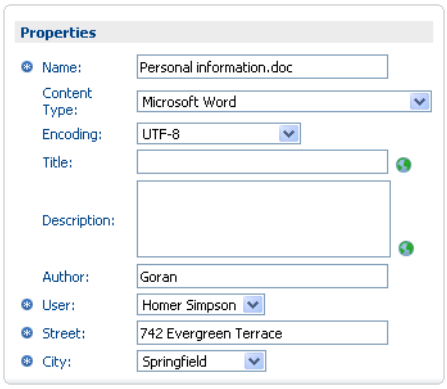

Figure 4. Visualisation in the edit mode

6. CONCLUSION

The Alfresco system provides support for the storage, management and retrieval of content. Content may range from coarse-grained documents to fine-grained snippets of information (such as XML elements). To describe the structure of such content, Alfresco supports a rich content model where the properties, associations and constraints of content are described. Out of the box, the Alfresco content model is pre-populated with definitions that describe common content constructs such as folders, files and metadata schemes. However, each Alfresco installation will have its own content requirements and as such the content model is extendable allowing the Alfresco to manage new types of content.

In this paper we propose the Alfresco extension that provides retrieval of data from external relational databases in a organisation and use of this data as metadata for folders, documents and business processes within Alfresco. In this way, a business information system of an organisation serves as a data source for some Alfresco metadata. The proposed solution can use different databases as a source.

A further direction in development of the module includes its extension to provide efficient support for cases when a mapped database table contains large amount of data. Also we plan to investigate cases when data is retrieved form two or more tables.

REFERENCES

[1] ISO IEC 82045-1, Document Management – Part 1: Principles and Methods, International Organization for Standardization, 2001.

[2] Shariff, M. *Alfresco Enterprise Content Management Implementation*, Packt Publishing, 2006.

[3] Alfresco Enterprise Content Management, http://www.alfresco.org

[4] NorthWest Computer Support, 5 Benefits of Using Document Management Systems (DMS) for Business, http://www.nwcsupport.com

[5] Alfresco Repository API Documentation, http://dev.alfresco.com/resource/docs/java/repository/

[6] Alfresco 2.0 Default Content Model UML, Alfresco,http://wiki.alfresco.com/wiki/2.0_Default_Content_Model_UML

[7] JSF API Specification, http://java.sun.com/javaee/javaserverfaces/reference/api/index.html

[8] Alfresco Developer Guide, http://wiki.alfresco.com/wiki/Developer_Guide

[9] Spring Framework Reference and API, http://www.springsource.org/documentation

[10] Mann, K., Java Server Faces in Action, Manning, 2005

ACKNOWLEDGMENTS

Results presented in this paper are part of the research conducted within the Grant No. III-43007, Ministry of Science and Technological Development of the Republic of Serbia.