IMPORTING DOCUMENT HIERARCHY IN THE ALFRESCO SYSTEM

Stevan Gostojić[1], Goran Sladić[1], Branko Milosavljević[1]
{gostojic, sladicg, mbranko}@uns.ac.rs
[1]University of Novi Sad, Faculty of Technical Sciences

Abstract *– This paper presents an extension of the Alfresco document management system that automatically imports document hierarchy templates from scripts. Although the similar solutions already exist, they are platform dependent and still in the development stage. The proposed solution uses platform independent Java application to remotely import document hierarchy into the Alfresco repository and can be easily extended to accommodate for different template formats and/or repository content models.*

Keywords *– Alfresco, document management system, extension, importing*

1. INTRODUCTION

A document management system is an information system that enables initiation, preparation, establishment, usage, revision, archiving and deletion of documents. In other words, it is an information system that is used for life-cycle management of electronic and paper-based documents.

A document can be defined as a fixed and structured amount of information that can be managed and interchanged as a unit between users and systems [1].

According to the same source, there are four types of documents: single documents, compound documents, document aggregations and document sets. Single document represents elementary form of information carrier in document management systems. It consists of the basic document and the associated metadata. Compound document is the result of the composition of multiple documents of various types. Metadata are assigned to the compound document as a whole, while its components do not have separate metadata. Document aggregation is a set of documents, each with its metadata. It has its own metadata, but does not need to have its own document. Document set is a set of documents with assigned metadata. While the set itself has assigned metadata, it does not have its own document. Furthermore, documents can be linked and can have different versions.

Each document is assigned metadata that describe its content or other characteristics. Document management systems can request users to enter metadata or can automatically extract metadata from its content [2].

The Alfresco [3] is the main part of the software component of a document management system. It uses open source technologies (it is licensed under GNU General Public License) and open standards. It can be relatively easily customized and extended by modifying XML documents, modifying source code or implementing web services that communicate with it. It can be also easily integrated with other systems using open standards.

While developing document management systems that are deployed in different organizations, authors have noticed that the automatic importing of document hierarchy from templates is one of the key functions that was missing. This paper describes an extension of the Alfresco, a specific document management system, that implements this function.

The rest of this paper is structured as follows. Section 2 reviews Alfresco document management system. Section 3 describes developed tool that imports document hierarchy into Alfresco repository. Finally, the last section gives concluding remarks.

2. ALFRESCO

Apart from document management, Alfresco supports web content management **(**managing files and content specifically intended to be delivered to the web), digital asses management (managing graphics, video, and audio), records management (managing content as a legal record) and imaging (capturing, tagging, and routing images of documents from scanners). Nevertheless, the attention is paid to its document management function.

This section reviews Alfresco document management system: its content model, functions, software architecture and technologies used in its development.

2.1. CONTENT MODEL

The Alfresco content model is used to represent types of content within its repository. The Alfresco content model is built using: types (used to model business objects, they have properties and can inherit a parent type), properties **(**represent metadata associated with a particular type), property types (describe the fundamental types of data the repository will use to store properties), constrains (used to restrict the values of a property), associations (used to define relationships between types, i.e. hierarchy of folders) and aspects (a mechanism to extend content model with properties and associations by attaching them to content types when and where they are needed).

One of the built-in content models is the Content Domain Model which describes Content, Spaces and predefined set of metadata. First two concepts roughly correspond to single document and document set respectively.

2.2. FUNCTIONS

Document life-cycle management is implemented with the support for different structural types of documents, as-

signing metadata to documents, definition and execution of business processes, document storing, indexing and retrieval, version control, security, document transformation, user collaboration and integration with other systems.

The predefined set of metadata can be easily extended and metadata can be automatically extracted from documents in different formats.

2.3. SOFTWARE ARCHITECTURE

Alfresco architecture is a typical three-tier internet architecture. It consists of user interface subsystem (Web Browser), business logic subsystem (Application Server) and data management subsystem (Storage). User interface subsystem interacts with the user. Business logic subsystem consists of Alfresco applications and the Alfresco repository. Alfresco applications are customized to specific areas of content management such as document management, web content management or records management. Alfresco repository provides a set of generic content management services such as storage, retrieval, version control and content transformation. The data management subsystem consists of the relational database management system that stores metadata and the file system that stores binary content. The architecture of the Alfresco system is shown in Figure 1 [4].

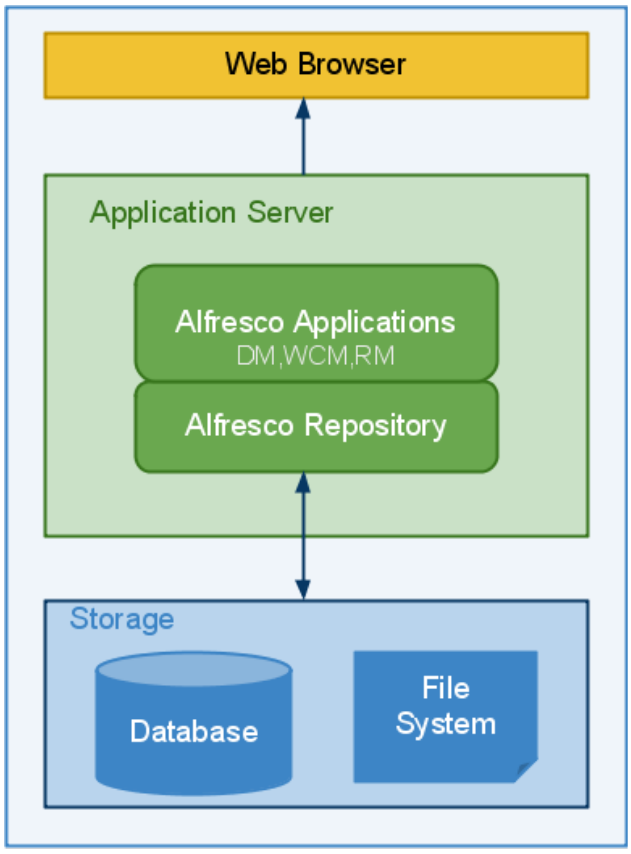

Figure 1. Alfresco Repository Architecture

2.4. TECHNOLOGIES USED

The system is implemented with JEE (Java Enterprise Edition) technologies which makes it independent from operating system, i.e. the system can be deployed to different operating systems that support Java platform (Windows, GNU/Linux, Mac OS, etc.).

JBoss or Apache Tomcat can be used as application servers under which Alfresco is run. It uses Hibernate object-relational mapping tool for access to relational database management systems. That makes usage of different relational database management systems possible. So far, the system is tested on MySQL, Oracle and Microsoft SQL Server.

It uses OpenOffice.org for transformation from one textual and hyper-textual document format to another and ImageMagic for transformation from one graphical document format to another.

As an user interface, the Alfresco can use a document management web application that can be accessed with a web browser such as Internet Explorer or Firefox. The web application is localized, among others, to the Serbian language.

Alfresco can also be integrated into Microsoft Office or OpenOffice.org office suites and the file system.

Finally, the repository can be accessed with the REST-style or the SOAP-style web services and, supposedly, with the JSR-170 compliant API over the RMI bridge.

3. CONTENT IMPORTER

This section describes the novel tool developed for importing specific content into Alfresco repository. It reviews existing solutions, presents requirements specification of the new solution, and describes design and implementation decisions.

3.1. EXISTING SOLUTIONS

Alfresco provides export and import functions out-of-the-box by providing corresponding repository services. Documents are exported to and imported from files in the ACP file format which is simply a ZIP file that contains an XML document that describes repository nodes and files that store binary content of documents.

Export and import services can be used by executing a repository action or by using export and import command line tools. Furthermore, Alfresco supports JSR-170 Level 1 and Level 2 specification which includes export and import functions.

Nevertheless, exporting and subsequent importing of content can only be performed between compatible versions of the Alfresco repository (which generally means the same version [5]). Although importing content into the repository that already contains content with the same UUID (Universally Unique Identifier) as the content being imported should be possible by specifying appropriate options, authors have noticed that this process is not working correctly.

There are some open source tools that automatically generate ACP files from directories in the file system or metadata in the CSV format and content stored in the file

system. Those tools are either platform dependent (they run only on the Microsoft Windows platform) [6] [7] or are still in the development stage [8].

Due to the described drawbacks of existing solutions, we have decided to design and implement our own solution which is described in following subsections.

3.2. REQUIREMENTS

The tool is required to generate space templates (the hierarchy of documents with assigned metadata) in the Alfresco repository from scripts in the CSV format.

It should be platform independent and use open standards where possible. Remotely generating space templates in multiple repositories and integration with command line scripts would be an advantage.

3.3. DESIGN AND IMPLEMENTATION

There are three methods to remotely access Alfresco repository: the SOAP-style web services API, the REST-style web services API and the JSR-170 Level 1 and 2 compliant API over the RMI bridge. The proposed solution uses SOAP-style web services as a communication mechanism between the client and the server because of interoperability issues.

The system is deployed on two nodes. The server node contains an application server that hosts the Alfresco repository. The client node contains the client application. Those nodes communicate by exchanging SOAP messages over the HTTP protocol. Deployment diagram of the system is shown in Figure 2.

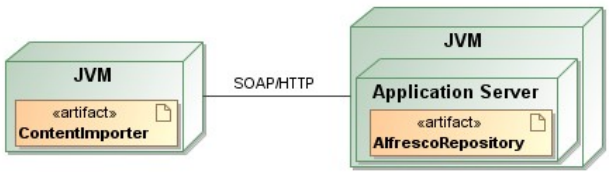

Figure 2. Deployment diagram of the system

The client application is written in the Java programming language in order to provide for platform independence. It is structured into packages. Class diagram of the system is shown in Figure 3.

The *Parser* package contains *Parser* and *CSVParser* classes. Those classes are used for parsing textual representation of the content into its object representation. The *Parser* class is an abstract class that is used as an interface for concrete classes. The *CSVParser* class parses the CSV (coma separated values) representation of content into its object representation.

The *Model* package contains *Content* and *Space* classes. Those classes correspond to entities in the Alfresco content model. They can be extended in order to accommodate for custom content types or aspects that are applied to the content.

The *Generator* package contains *Generator* and *ContentDomainGenerator* classes. Those classes generate content in the remote repository from its object representation. The *Generator* class is an abstract class that is used as an interface for concrete classes. Its subclasses use *AuthenticationService*, *RepositoryService*, *ContentService*, *ActionService* and *AccessControlService* classes that are proxies for the corresponding repository web services. The *AuthenticationService* is used for the user authentication. The *RepositoryService* is used for the repository manipulation. The *ContentService* is used for reading and writing content. The *ActionService* is used for adding content rules (repository actions that will be automatically triggered when content with specific properties is added to the repository). The *AccessControlService* is used for the management of access permissions. Those classes are provided by the Alfresco project and are contained in their own package. The *ContentDomainGenerator* class generates content for Content Domain Model.

Finally, the *ContentImporter* class interacts with the user using the command line interface and connects all the pieces of the application together.

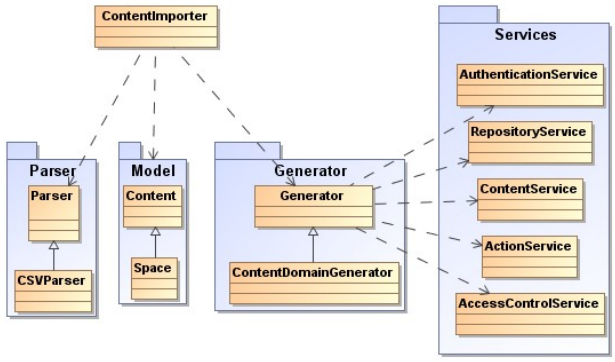

Figure 3. Class diagram of the system

Sequence diagram of the system is shown in Figure 4. Firstly, the textual representation of content is parsed into its object representation. Then, the session to the Alfresco web service is opened. The rest of the web services mentioned in this article send the session identifier in the SOAP message header. Each document in the document hierarchy represented by its object representation is added to the repository by creating content of the specified type with specified aspects applied, writing document content to the repository, saving all rules that are connected with the document and setting its access permissions. Finally, the session is closed.

4. CONCLUSION

The Alfresco system is an extensible solution to document management based on the open source technologies. There are existing solutions that support content importing from templates, but those solutions are platform dependent and/or in the development stage. That is the reason why the Alfresco system has been extended in order to remotely generate content templates from scripts. Developed solution is platform independent, can be easily extended to accommodate for different template formats and/or repository content models and can be used in the command line scripts.

Possible direction of further development is a parser that parses ACP files that can be used within importing tool.

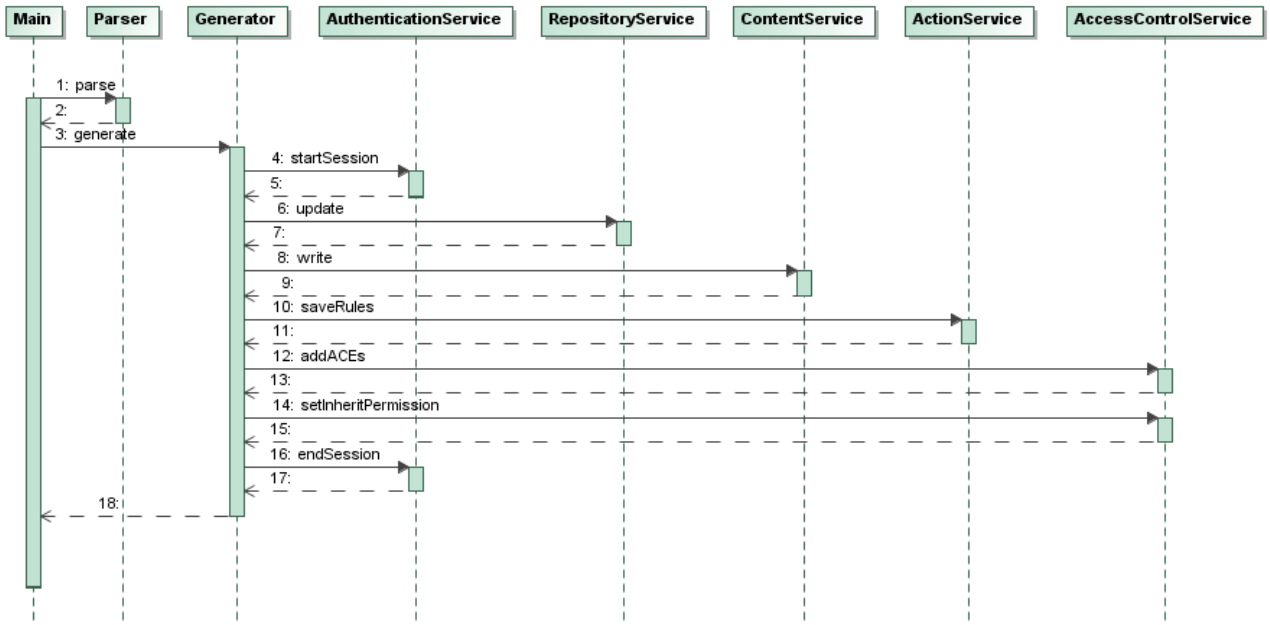

Figure 4. Sequence diagram of the system

REFERENCES

[1] ISO IEC 82045-2: *Document Management – Part 1: Principles and Methods*, International Organization for Standardization, 2004.

[2] Milosavljevic B. et al., *Project: Requirements Specification of the Intranet System of the Pariliamant and the Cabinet of the Authonomous Province of Vojvodina*, Faculty of Technical Sciences, Novi Sad, 2006.

[3] Shariff M., *Alfresco Enterprise Content Management Implemantation*, Packt Publishing, Olton, 2006.

[4] *Alfresco Repository Architecture*, http://wiki.alfresco.com/wiki/Alfresco_Repository_Architecture (accessed on 27th of January 2011)

[5] *Export and Import*, http://wiki.alfresco.com/wiki/Export_and_Import (accessed on 27th of January 2011)

[6] *Alfresco Forge: ACP Creator*, http://forge.alfresco.-com/projects/acpcreator/ (accessed on 27th of January 2011)

[7] *Alfresco Forge: APC Generator*, http://forge.alfresco.-com/projects/acpgenerator/ (accessed on 27th of January 2011)

[8] *Alfresco Forge: APC Generatr*, http://forge.alfresco.-com/projects/acpgeneratr/ (accessed on 27th of January 2011)

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.