

The SOA Magazine

Feature Article



A Content Management Migration Framework for SCA

by Gregory Green

Published: April 30, 2009 (SOA Magazine Issue XXVIII: April 2009)

[Download PDF](#)

[Digg This](#) • [De.licio.us](#) • [Slashdot](#) • [Technorati](#) • [StumbleUpon](#) • [Google Bookmark](#)

Abstract: Migration tools are typically hampered by issues related to their complexity, slow processing and inability to meet custom migration requirements. In many cases, a large amount of time and effort is spent to create these solutions that are disbanded after completion. "Reuse" - a core principle of service-orientation - is rarely thought of in these efforts although many other migration efforts will need to solve some of the same problems. There is a major need to have a reusable approach that is fast and easy to apply. It should also provide the ability to provide custom functionality for all aspects of migration processing.

Customers typically want to create a single repository for all content; thus providing a single point of access, configuration and management. A simplified process that is flexible enough to comply with a wide range of Standard Operating Procedures (SOP) along with legal/regulatory requirements is essential.

Migration from older systems to an enterprise content management system may be needed for enhanced document search features, version control, security, life cycle management and workflows. The software that facilitates the migration must be powerful, flexible, fast and accurate. This will reduce the size of the migration staff, shorten the timeframes, minimize errors and ultimately lower the overall migration cost.

A key goal of this approach is to provide the ability to migrate any type of source or target system. The approach must supply "out of the box" functionality that will support most of the migration needs. It should also act as a "building block" to fill-in any missing pieces.

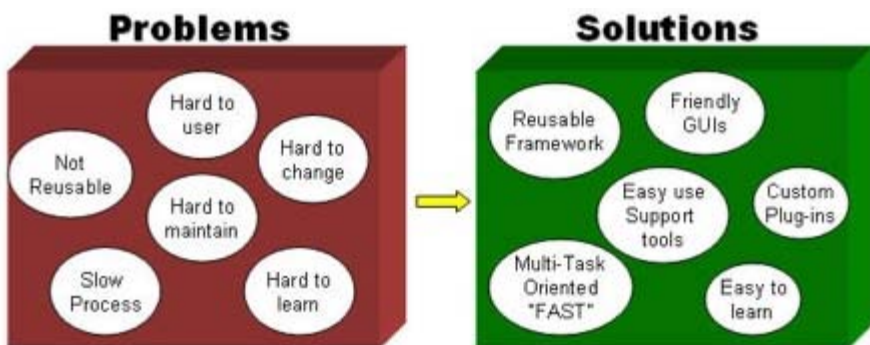


Figure 1

Introduction

This article provides a design approach for a content management migration. The goal of this approach is to produce an effective process for migrating content from one system to another. It explains concepts needed to support a smooth and successful transition.

Typically, there are four stages in the migration process; collection, translation, validation and importation. The classic content entities are documents, folders, users and groups.

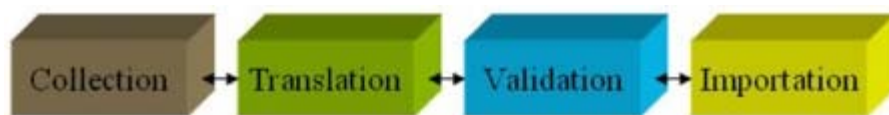
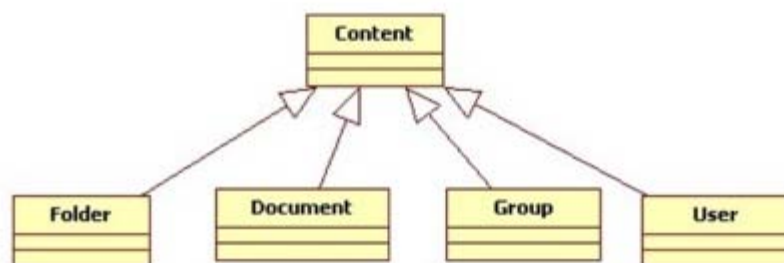


Figure 2



Collection

The term collection refers to the ability to gather information from a target system. The collection information is stored in a temporary data store (also known as a staging area) in a structured manner. Collection components may be developed to support content collection from the content server such as Alfresco Content Management Server and Live Link. The staging area contains extracted content along with meta-data) may be a combination of CSV, XML, MS-Access, Open Office ODF file, RDMS or any other structured data storage mechanism.

Translation

The collected data may need to be translated to conform to the targeted content management system's requirements. The ability to transform/alter collected attributes (also known as meta-data) or content is based on configured translation rules. The approach should support translation processing as part of the data collection or may be executed separately. Translation is typically configured as a XML or MS-Excel file. It defines how attribute of a source system must be modified or defined within the target system. Translation may use a framework like Saxon XSL or a scripting language.

Validation

Prior to importing content, the inputted data needs to be validated. For example, prior to importing a single document, the process should validate that the folder where the document will be placed or a referenced document owner exists in the target system. This approach should support validation as part of content import or as a separate process. Validation (just like translation) should be based on configuration validation rules. Validation may be implemented using any validation framework such as OVAL, spring validation and struts.

Importation

Import refers to the ability to upload content/entities to the target content management system. Imported content along with their attributes are typically taken from the output of the collection and translation in the staging area.

The import process must support simplified remediation of content. Remediation refers to the ability to identify and correct issues that occur during importation. Remediation is typically a manual process. The importation tool should produce an error log of issues that have occurred. The error log format should mirror the import input format with additional information that describes the nature of the error. This would allow the migration specialist to correct the issues and rerun the import process with the error log as its input. The migration specialist may continue to rerun the import process error logs until all issues are resolved and all content is imported.

Importation into enterprise CMS(s) such as Documentum Content Server and SharePoint should be supported.

Migration Architecture

This section gives an architectural approach for the content migration. The following diagram illustrates the key components of the approach.

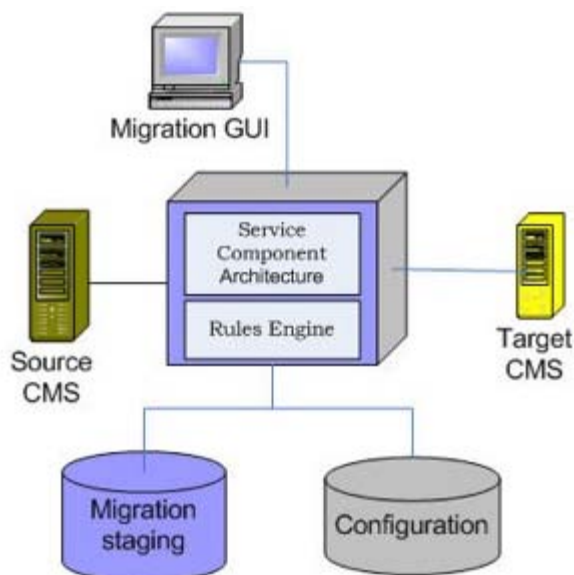


Figure 3: Conceptual Architecture

Migration GUI

The migration specialist should use a rich client Graphical User Interface (GUI) to control the migration processes. The GUI will be a client of the core services. It acts as the director of all migration activities. It will allow users to specify the tasks for each migration step. It will allow the migration specialist to view the migration process (progress bars will be used whenever possible) and analyze results. This GUI should provide controls that are easy to use. The migration specialist can execute the entire migration or certain parts of it such as collection, translation, validation or importation. The migration specialist always knows the status of each operation with the progress bars. Java FX, Swing or Adobe Flex would be ideal technology choices to build this type of front end.

Core Services (SCA)

The approach's core services should be built on a Service Component Architecture (SCA) platform along with a Rule Engine. SCA is a standardization of service-oriented architecture concepts for application development. The Rules Engine should support the ability to configure and execute both translation and validation rules. The management of migration should be controlled by a multi-threaded server process. This server should expose remote interfaces to start, stop and obtain progress information.

The main goal of SCA is to decouple migration logic from the details of its service operations. With SCA, the actual target services can be any program language including Java, C#, and Ruby as well as XML, BPEL, and XSLT. Service Component Architecture is based on the industry standard service-oriented architecture. SCA components can communicate with each other using One-Way, Asynchronous, Call-Return, and Notification. Each SCA platform supports Security, Transactions and Reliable Messaging. Building on top of an SCA platform will improve the overall flexibility and reusability of the platform.

Each functional module within the platform can be represented as an SCA component. A component may implement one or more services. In terms of SCA, a service can be thought of as a programming language interface definition. Components may have one or more references to an external service. These references are themselves implemented by other components. A major benefit of this architecture is that it is loosely coupled. Components do not directly reference each other. Instead they reference the service interfaces. The implementations of these interfaces are wired together by the SCA platform. This allows components to be swapped in or out very easily. The service implementation and references are configured during

deployment and injected at runtime (dependency injection). Since interactions highly rely on service interfaces, a key success factor is to design them to be flexible and less likely to change over time.

The components are grouped into an SCA composite. For example, the Collection composite contains the Document, Folder, User and Group collectors. In Figure 2 Framework Composite/Component View the boxes named "Collection", "Translation", "Validation" and "Importation" represent the composites. The blocks inside the composites are the components.

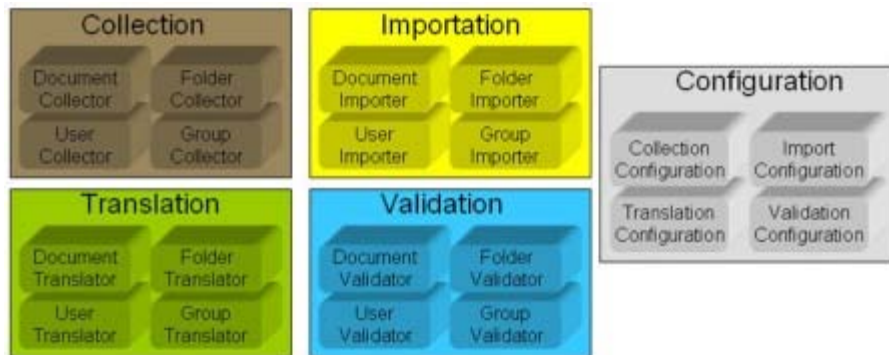


Figure 4: Framework Composite/Component View

CMS

The content migration approach is designed to extract from a source CMS migration system and put it into a target CMS system. The source and target CMS(s) must provide API(s) and user interfaces to support content creation, read, update and delete operations.

Source

Content is extracted from a source CMS by components in the Collection composite. Each collector (in the Collection composite) uses the API/integration interfaces of the source CMS to export the content data along with its meta-data into a temporary storage area (migration staging data-store).

Target

Content is imported into a CMS by components in the Importation composite. Each import component uses the CMS import interfaces to add or update users, groups, folders and documents. The content (along with its meta-data) is read from the staging data-store and then upload to the Target CMS.

Configuration

Configuration of the platform should rely on SCA and rules engine configuration support. Each component can be configured using an XML file. Each component may expose various properties to be configured through SCA configuration interface.

Migration Staging

Translation, validation and importation are all performed on data in the migration data-store (staging). This staging data-store may be local files on a server's file system, a Relational Database Management System (RDMS) or a combination of both.

Conclusion

The approach is designed to solve the typical issues that hamper most migration efforts. There are many platforms or implementations that may have been chosen, but the architectural principles are the same.

The GUI's ability to display the migration progress can provide improved usability. SCA promotes flexibility and reusability. A rules engine provides a way to introduce business logic into the migration along with enforcing a wide range of constraints.

Migration developers can add custom collectors or importations for any CMS. This is similar to a plug and play model. A CMS server that is not supported ("out of box") can be plugged into the approach once the construction of the necessary components that implement the service interface is developed. The migration specialist uses the same conventions to configure and execute the migration regardless of the configured collectors or importers.

This approach will address most migration efforts in a way that is adaptable and powerful. Most migration tools focus on a particular target or source system. This approach is generic and is targeted to work with any CMS server regardless of the supported interface programming languages, platforms or conventions. As long as the target and or source system supports the basic entities (such as the users, groups, folders and documents) then the content should be migrated. The approach provides a robust, flexible and simple way to meet most migration needs. The approach is a building block that can be reused time and time again; thus saving time and money.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About](#) [Legal](#)

Copyright © 2006-2009
SOA Systems Inc.