Real-World Challenges in a Multilingual and Multi-Locale Web Content Management Project

Christian Donner
43 Bancroft Road
Stoughton, MA 02072 USA

***Abstract*- Content management implementation projects can face both technical and non-technical challenges that are specific to Internationalization and Localization requirements. Technical challenges can be a threat to a project's budget or schedule, but can usually be overcome with technical solutions. Non-technical challenges are more difficult to spot, to understand, and to resolve, and are therefore a more serious threat that has the potential to derail such a project completely. With proper planning and preparation and, hopefully, with the help of some of the ideas presented in this document, your international CMS implementation will be successful. You will learn about strategies for content authoring, content management and content translation, but also find solutions to specific technical road blocks.**

I. Introduction

The web solution described in this paper was built for an international publisher of financial research and analysis. The publisher covers global and local markets. Prior to this initiative, the company operated a large corporate site and a series of smaller product sites and regional sites that were all not content-managed. Information was difficult to find and the cost of maintaining 12,000+ static HTML documents grew rapidly. A Content Management System (CMS) was purchased (Fatwire Content Server) and a consulting firm was tasked with redesigning the web presence, streamlining the publishing process, implementing the product and building a multi-lingual site.

The decision for building one site supporting multiple locales was supported by three factors:

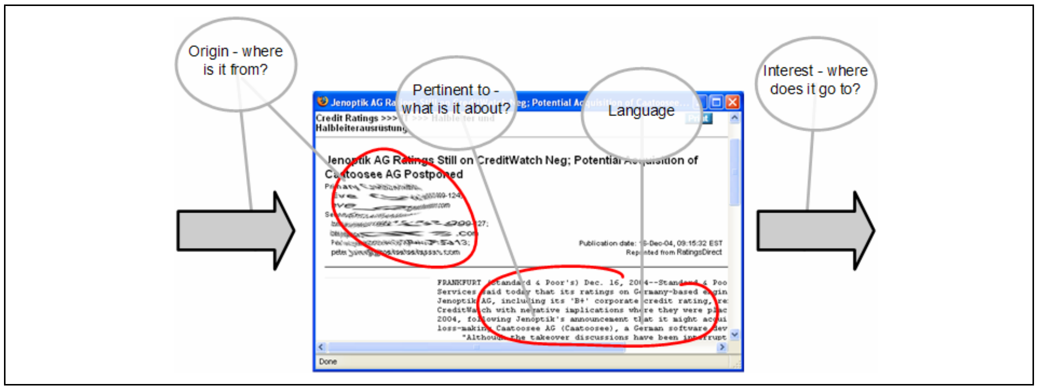
- Cost: A web site consolidation project was expected to help centralizing content assets and reducing the cost of maintenance
- Value: Self-publishing and workflow was expected to reduce the time-to-market for new content
- Opportunity: Strong economic growth in Asia required an investment to better cover the financial markets in this area

This paper describes the concepts that were developed and used in support of this initiative.

II. International Content Classification

International content be described by using a set of 4 attribute classes (see Figure 1):

- Content can originate **from** a certain region, e.g. an article is published by the Frankfurt office.
- Content can contain information **pertinent to** a certain geographical region, for instance a German company.
- Content can be **of interest to** users in one or more regions, for instance Europe and the USA.
- Content can be written in one or more **languages**, e.g. German and/or English.


Origin - where is it from?
Pertinent to - what is it about?
Language
Interest - where does it go to?
Credit Ratings >>>
Halbleiterausrüstung
Print
Done

Jenoptik AG Ratings Still on CreditWatch Neg; Potential Acquisition of Caatoosee AG Postponed

Publication date: 16-Dec-04, 09:15:32 EST
Reprinted from RatingsDirect

FRANKFURT (Standard & Poor's) Dec. 16, 2004--Standard & Poo
Services said today that its ratings on Germany-based engin
Jenoptik AG, including its 'B+' corporate credit rating, re
CreditWatch with negative implications where they were plac
2004, following Jenoptik's announcement that it might acqui
loss-making Caatoosee AG (Caatoosee), a German software dev
"Although the takeover discussions have been interrupt

Figure 1 - Article attributes

This classification model turned out to be too complex for the client's needs. It was simplified by assuming that an article about Asia is of interest to the Asian region. This eliminated one degree of freedom. The origin of an article was not relevant for the site taxonomy[1], leaving only two attributes that content editors had to provide: region and language.

III. Content Targeting – Who Sees What

The segmentation of the web site's target audiences into regions and languages was the most important step of the analysis process. This client operates offices on all continents. Only the larger offices, however, contribute content to the web site. They are depicted on the map in Figure 2.

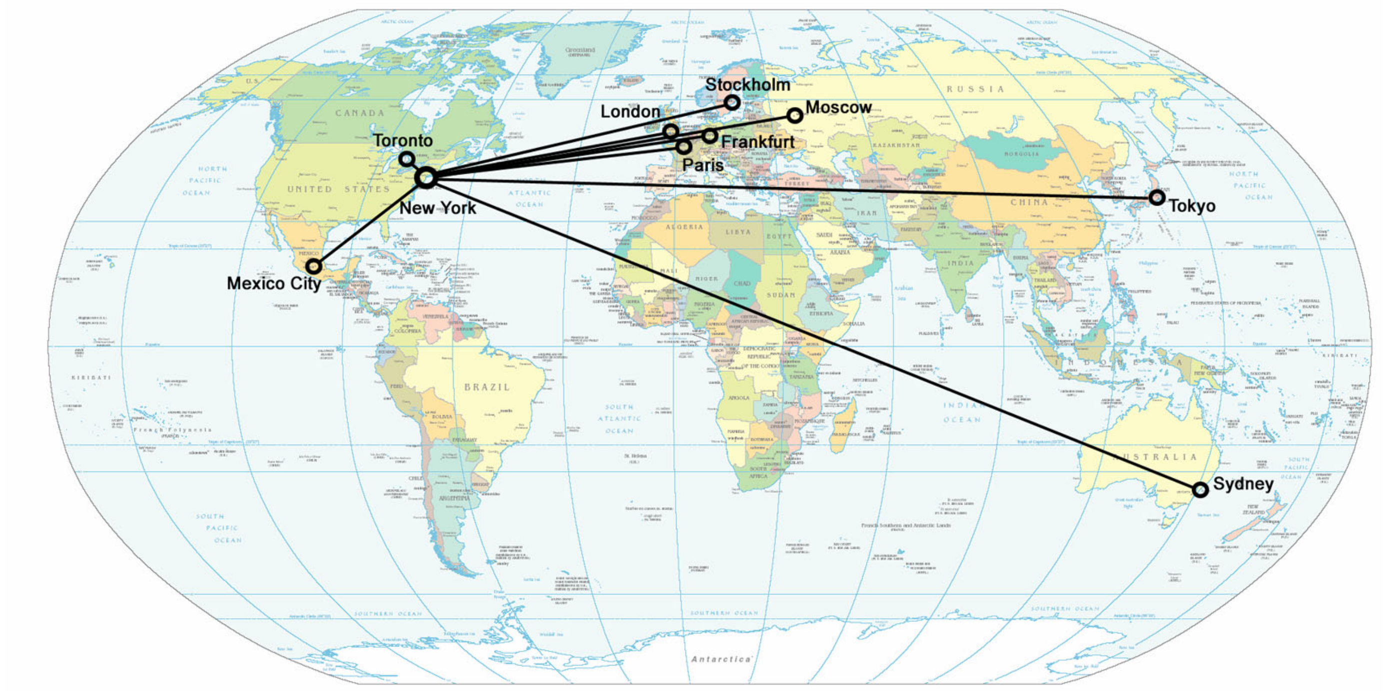


Figure 2 - Location of offices with content contributors

Because the client's target markets did not correspond to countries, the concept of tagging content with a locale code, comprised of a country code and a language code, did not suffice. Instead, markets corresponded to regions. Any content asset had to be characterized by its language, the region(s) that it applied to and, to enable searches by country, any number of countries within that region. For instance, Europe, the Middle East, and Africa (EMEA) were considered one region called "Europe". Each market or region was editorially covered in one or more languages. After several iterations of defining the regions and the languages that should be available, the following list was defined by the client:

► Global / English	► 日本/日本語 (Japan / Japanese)
► US / English	► Australia & New Zealand / English
► Canada / English	► Asia / English
► Europe / English	► Latin America / English
► Europe / Français	► América Latina / Español
► Europa / Deutsch	► América Latina / Português
► Europe (Nordic and Baltic) / English	► Россия; СНГ/Русский (Russia / Russian)
► Japan / English	

The Global region was a separate market and the other regions did not simply roll up to Global. To further complicate things, the client wanted certain English articles to appear on non-English sites. This was supported by the argument that most German-speaking site visitors also read and understand English and that the benefit of finding English content of interest would outweigh any other potential disadvantages for the user. For instance, a user who selected *Europa / Deutsch* will actually see some English content. Figure 3 illustrates the regions that the London office can publish content to.

[1] Taxonomy is a term used by information architects and describes the organization of the content on a web site.

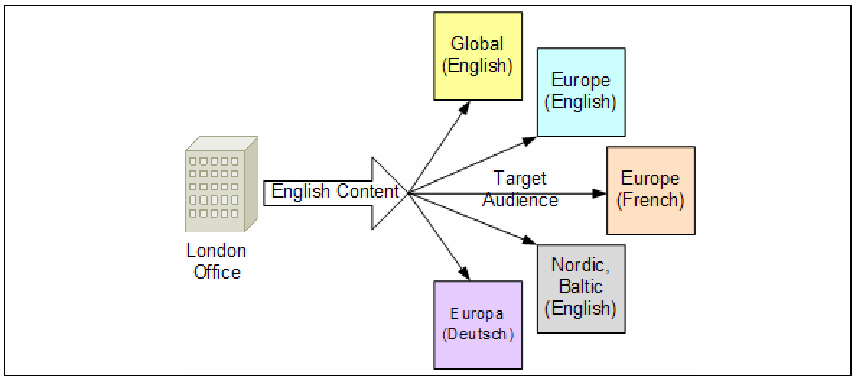


Figure 3 - Content creation for multilingual audiences

A solution was required that allowed a CMS user to publish an article to any number of regions. There are two fundamentally different ways of approaching this problem that will be discussed in the following chapter.

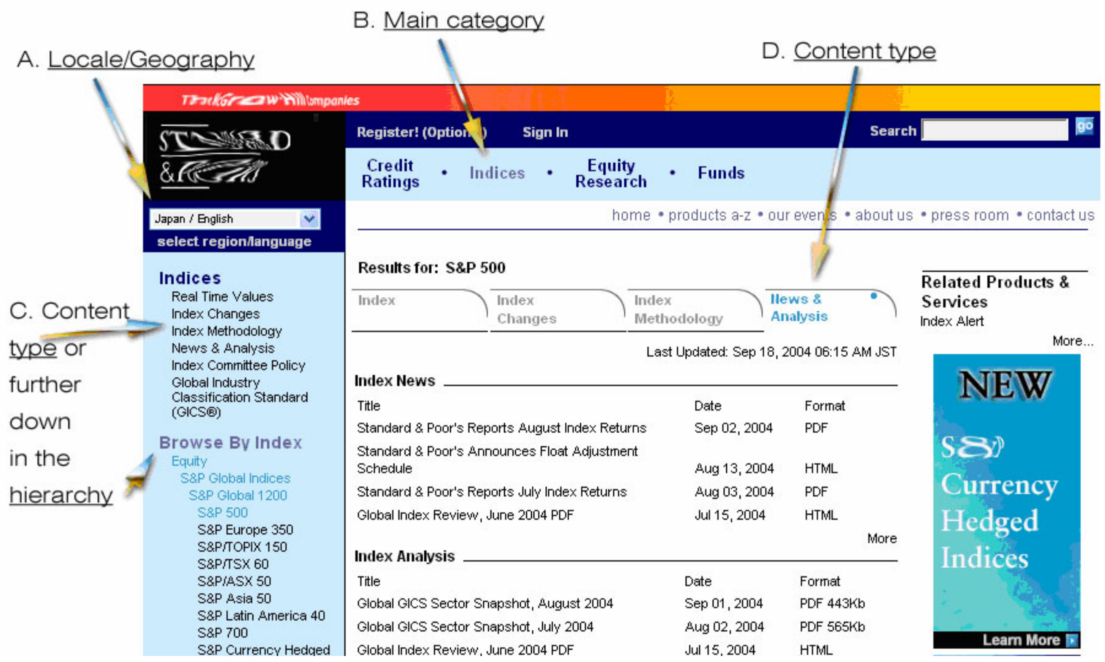
IV. Targeted Vs. Placed Content

A. *"Placed" Content*

In order to place content on a specific location of a web site, the content manager must provide additional information for the content asset. The asset must "know where it belongs". This can be done by storing a reference to the site location with the content asset in the database. For instance, the content manager would select a destination page from a drop-down list while he creates or edits the asset.

Since most real-life sites require that a content asset can be placed in more than one location, a many-to-many association is necessary. Technically, this is a table that contains references between content assets and taxonomy categories (taxonomy categories map to pages on the site and provide a layer of abstraction between the site navigation and the site taxonomy).

Taxonomies can be complex and there can be multiple taxonomies that overlap each other or apply orthogonally to a given site, as the example in Figure 4 shows. Taxonomies are likely different for some locales, as not all content categories apply to all regions. Users often have to navigate through several levels of the taxonomy and they can arrive at a certain content asset through different paths.



Indices
- Real Time Values
- Index Changes
- Index Methodology
- News & Analysis
- Index Committee Policy
- Global Industry Classification Standard (GICS®)

Browse By Index
- Equity
  - S&P Global Indices
    - S&P Global 1200
      - S&P 500
      - S&P Europe 350
      - S&P/TOPIX 150
      - S&P/TSX 60
      - S&P/ASX 50
      - S&P Asia 50
      - S&P Latin America 40
      - S&P 700
      - S&P Currency Hedged

Results for: S&P 500

Last Updated: Sep 18, 2004 06:15 AM JST

Index News

Title	Date	Format
Standard & Poor's Reports August Index Returns	Sep 02, 2004	PDF
Standard & Poor's Announces Float Adjustment Schedule	Aug 13, 2004	HTML
Standard & Poor's Reports July Index Returns	Aug 03, 2004	PDF
Global Index Review, June 2004 PDF	Jul 15, 2004	HTML

More

Index Analysis

Title	Date	Format
Global GICS Sector Snapshot, August 2004	Sep 01, 2004	PDF 443Kb
Global GICS Sector Snapshot, July 2004	Aug 02, 2004	PDF 565Kb
Global Index Review, June 2004 PDF	Jul 15, 2004	HTML

Related Products & Services

Index Alert

More...

Figure 4 - Primary navigation path of the site

In the example in Figure 4 the user arrived on this page by first selecting a locale (Japan/English), then a main category (Indices), then by drilling down 3 levels into Equity (Global Indices – Global 1200 – S&P 500) and finally by selecting the type of content he is interested in (News & Analysis). The content on this page is a subset of the News & Analysis that would appear had he selected News & Analysis at the top of the left-hand menu – an alternate navigation path.

Here is a look at a concrete scenario. The content manager created an article with the ID "001". The title is "Message from the CEO". The content manager would like to place this article on the Corporate News section of his site. He creates a row in the Article/Taxonomy association table that links article 001 to Category "Corporate News". In the taxonomy table, this category has the ID "537". Figure 5 shows the database relationships.

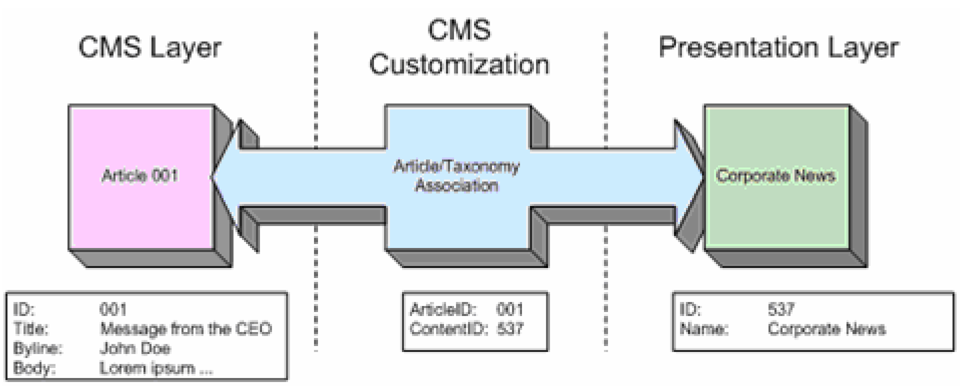
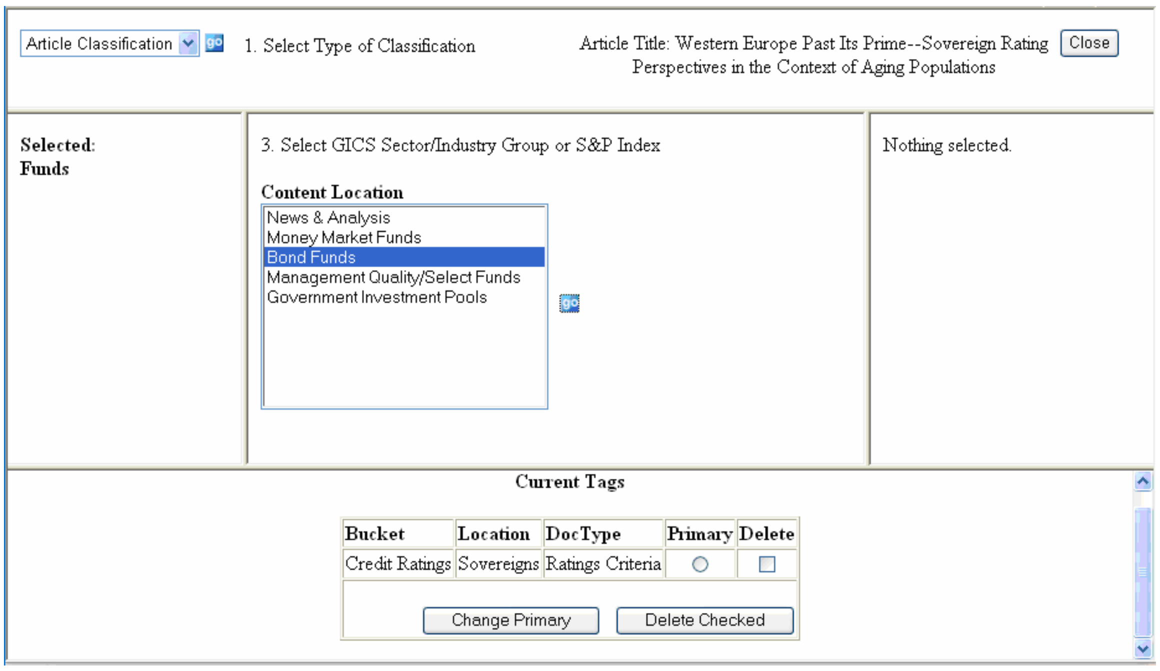


Figure 5 - Placing content through association

Because Content Server did not offer such functionality out-of-the-box, its user interface had to be customized. Figure 6 and Figure 7 show examples of this customization. The article classification screen requires 3 steps to be completed. The user is currently looking at step 2. The content of each list box depends on the previous selections. Only valid combinations are displayed.



1. Select Type of Classification

Article Title: Western Europe Past Its Prime--Sovereign Rating Perspectives in the Context of Aging Populations

Selected:
Funds

3. Select GICS Sector/Industry Group or S&P Index

Content Location

News & Analysis
Money Market Funds
Bond Funds
Management Quality/Select Funds
Government Investment Pools

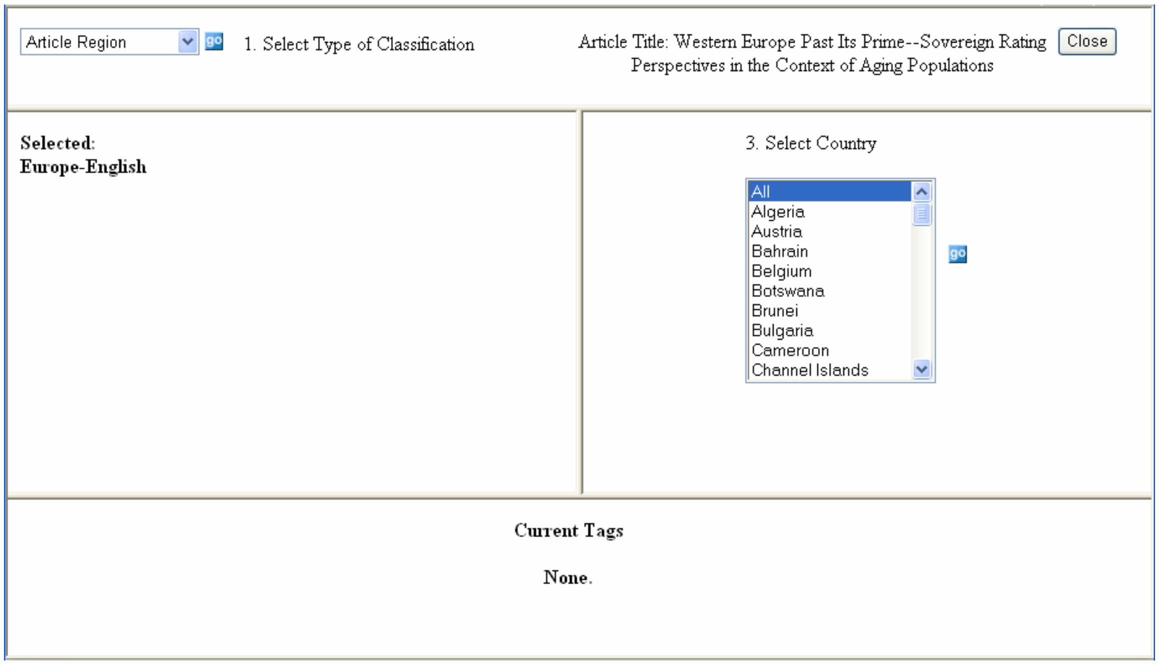
Nothing selected.

Current Tags

Bucket	Location	DocType	Primary	Delete
Credit Ratings	Sovereigns	Ratings Criteria	○	☐

Figure 6 - Site section and content type association

The region selection consists of 2 steps. First, the user selects a locale (a region/language combination that the site supports) and then (optionally) one or more countries. This allows site users to search for content that is relevant to their geography. Cameroon is a valid selection because "Europe" actually refers to the EMEA market, as mentioned earlier.



Article Title: Western Europe Past Its Prime--Sovereign Rating Perspectives in the Context of Aging Populations

Selected:
Europe-English

Current Tags

None.

Figure 7 - Locale and geography association

B.“Targeted” Content

Targeted content, on the other hand, is not directly associated with a taxonomy node or a navigation target, such as a page of the site. Instead, the content manager provides metadata information about the content in the form of labels, or keywords. One or more additional columns must be added to the content asset’s database table so that the keywords can be stored and retrieved.

The presentation layer must contain logic that can retrieve content containing keywords that match a given navigational node. For instance, to display content in the Corporate News section of a site, the query may be looking for assets that contain the keyword “Corporate”, as indicated in the example in Figure 8. Keywords change over time and should not be hard-coded into the presentation layer or the database query, of course.

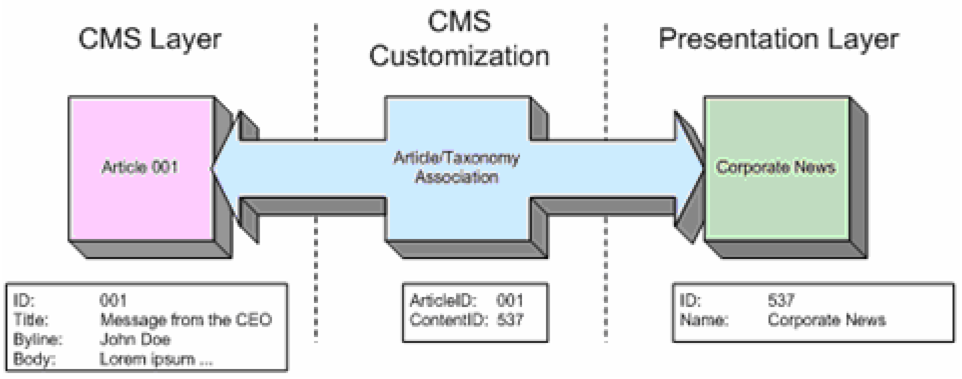


Figure 8 - Targeting content through keywords matches

The content manager in the above example can pick the keyword “Corporate” from a list while editing the asset. A content manager should not be allowed to freely type in keywords that are essential for the correct behavior of the site, because this allows misspellings of keywords, leading to content orphans that will never get displayed. Keywords must be tightly controlled in order to keep their number manageable and to ensure that there is only one form of spelling in the system.

In certain situations it is possible to store a list of keywords that are separated by delimiters. If the number of keywords for an asset becomes large, the list of keywords that the presentation layer is looking for is large. On sites that contain a large number of assets, parsing the keyword lists may become a performance bottleneck. In such a situation it is a good idea to store keywords in separate tables and establish many-to-many associations between the keywords and the content assets and the keywords and the taxonomy.

For content that comes from external sources, such as syndicated news feeds or product data provided by wholesalers, it is not possible to add placement information on the fly, and the use of the content metadata is the only way to find the correct location on the site.

As a side effect, keywords can be reused to appear in the META tags of the pages displaying the content. Although largely ignored by today's search engines, it is still a good practice to provide META tags and your internal site search engine can use this information to better rank search results.

C. Combining "Placed" and "Targeted" Content

For this project it was necessary to implement a combined model, allowing content managers to target content to a certain section of the site and labeling content for application-driven targeting at the same time (Figure 9).

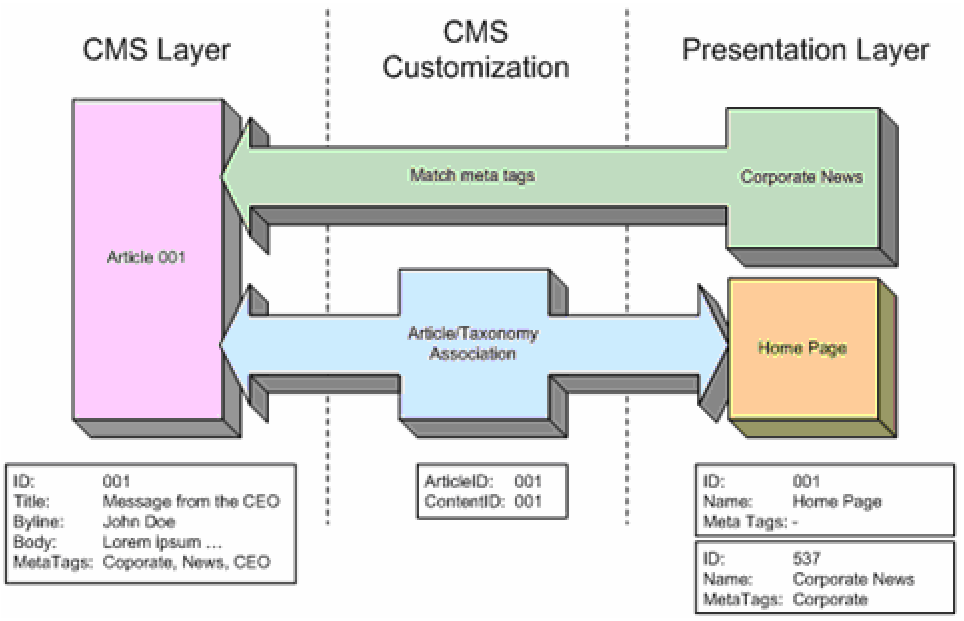


Figure 9 - Hybrid content selection

For instance, content managers may wish to hand-pick content that appears on a site's home page, while they feel confident about automating the display of content in the Corporate News section through the use of keywords. The example above shows how this would be reflected in the data.

It is possible for a site implementation to include both concepts because they work separately and independently.

V. Date Format Requirements

A common standard for date formats was among the most difficult things to achieve in this project. Consequently, we had to change the site's date formatting behavior several times. Initially the client favored the implementation of the US date format for all locales, but later experienced resistance from the regional offices. We then localized the date display. The format also changed from full month names to 3-letter abbreviations.

The Content Server administration user interface was customized to allow content contributors to enter local date and time values, but all dates were converted to Eastern Standard Time (EST) before they were stored in the database (Oracle 8i did not support time zones – broader time zone support was added in 9i [2]). On the display side, a JSP custom tag was used to convert the EST date and time values back to the localized format.

VI. Java Code Internationalization

Resources for code internationalization are available on the Web in large numbers. Good examples are [4], [5] and [6]. The following is a fairly complete checklist of things that need to be addressed in a Java internationalization project.

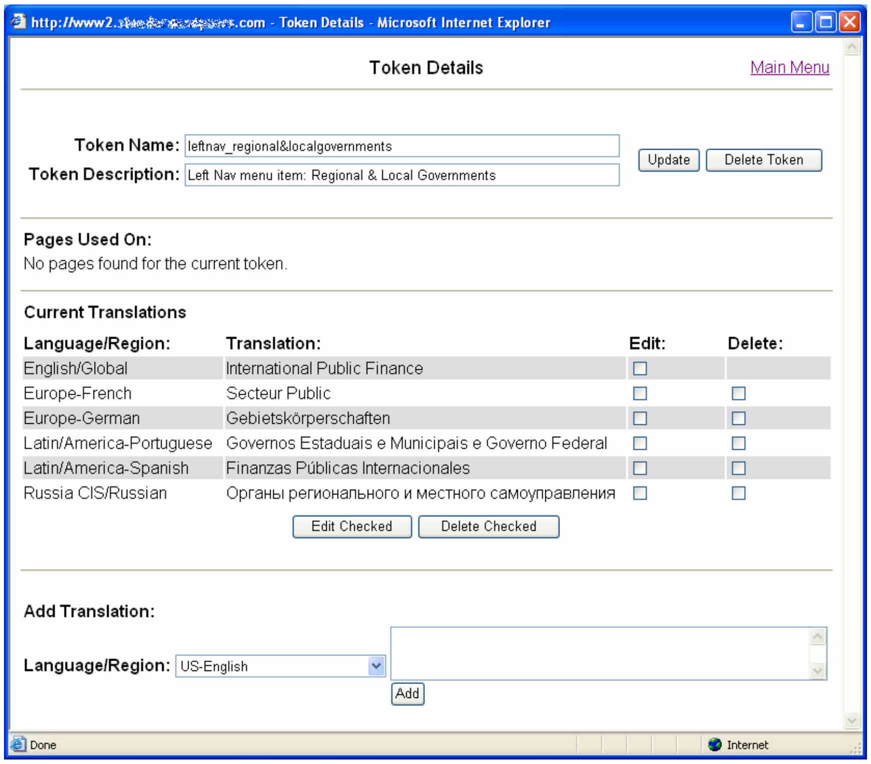
- Messages
- Labels on GUI components
- Online help
- Sounds
- Colors
- Graphics
- Icons
- Dates
- Times
- Numbers
- Currencies
- Measurements
- Phone numbers
- Honorifics and personal titles
- Postal addresses
- Page layouts

Developers had a checklist of required practices that worked well in general, but specifically the amount of work required to clean up the page layout for all languages was underestimated. Once the translations for all the menu items and the taxonomies for the regions were finalized, it turned out that the pages had to be reworked manually, because menu items wrapped in the wrong places or pushed out table cells to the right. There is no way around this issue. We experimented with different menu layouts, fonts, and sizes, but none of the templates developed worked without issues for all problems.

VII. Translation Process Facilitation

A one-pass translation is one possible approach for the localization of labels and headlines on the pages of the site. For example, a document would be created that contains the label names (tokens) and the root translation (in this case, the translation root was the one for the Global/English region). The document would be sent out to the regional offices for translation. The regional offices would then assign someone to add the translations for their regions to the document.

This process seemed to be inadequate, because the tokens would appear out-of-context and in many cases their original meaning could be misinterpreted. Instead, the project team built an online tool that allowed the translations to be done in a continuous fashion, thus allowing for ongoing improvements to the quality of the site. The following screen shows the functionality of this tool.


http://www2.[redacted].com - Token Details - Microsoft Internet Explorer

Token Details

Main Menu

Token Name: leftnav_regional&localgovernments

Token Description: Left Nav menu item: Regional & Local Governments

Update | Delete Token

Pages Used On:
No pages found for the current token.

Current Translations

Language/Region:	Translation:	Edit:	Delete:
English/Global	International Public Finance	☐	
Europe-French	Secteur Public	☐	☐
Europe-German	Gebietskörperschaften	☐	☐
Latin/America-Portuguese	Governos Estaduais e Municipais e Governo Federal	☐	☐
Latin/America-Spanish	Finanzas Públicas Internacionales	☐	☐
Russia CIS/Russian	Органы регионального и местного самоуправления	☐	☐

Edit Checked | Delete Checked

Add Translation:

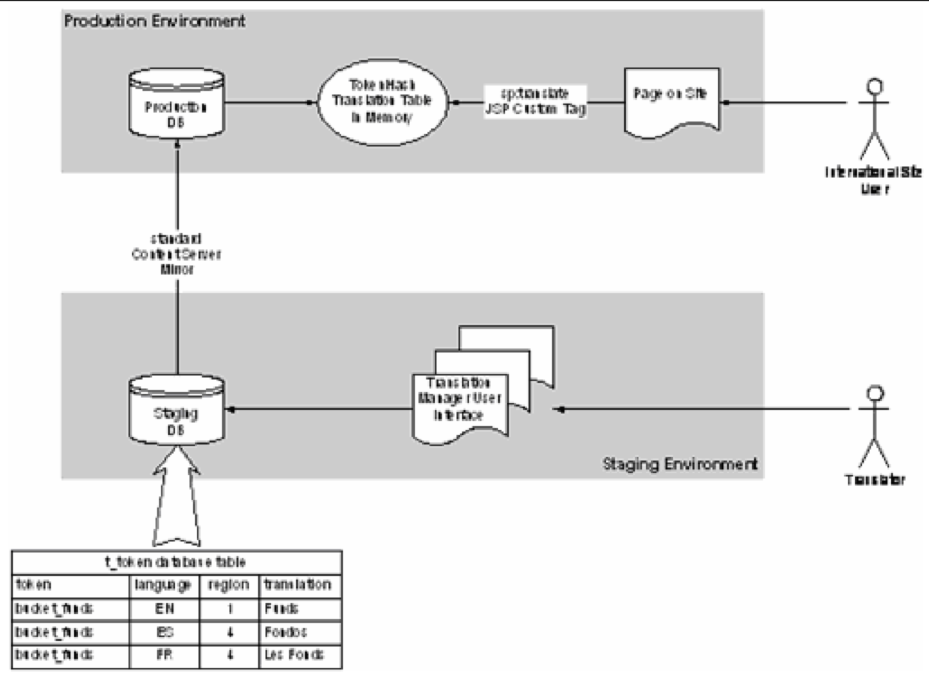
Language/Region: US-English

Add

Done — Internet

Figure 10 - Translation Manager

Authorized users were able to browse and search for tokens. Token details are displayed on this page in Figure 10. A nomenclature was chosen so that tokens could easily be found by the translators and editors. For instance, all elements of the left navigation contained the prefix *leftnav_*. Users could then edit the token description and add, modify and delete translations for language/region combinations. If no translation was provided for a given language and region, the site displayed the root translation.


Production Environment
Production DB
TokenHash Translation Table In Memory
sp:translate JSP Custom Tag
Page on Site
International Site User
standard ContentServer Mirror
Translation Manager User Interface
Staging DB
Staging Environment
Translator

t_token database table			
token	language	region	translation
bucket_funds	EN	1	Funds
bucket_funds	ES	4	Fondos
bucket_funds	FR	4	Les Fonds

Figure 11 - Translation Process

Because translators work in a staging environment that is separate from production, their new translations became not immediately visible on the site. Content Server's built-in mirror publishing functionality was used for scheduled change propagation of the token table to the production environment. Figure 11 above illustrates the interactions of the staging and the production users with the system.

Because tokens were created in several different ways – manually by the developers as needed for a page and in bulk from data – the token table eventually contained a sizeable number of duplicate entries for text elements that appeared frequently on the site. This became a problem, because translators eventually came up with different translations for the same text. Because this was a systemic issue, we were not able to eliminate the creation of duplicates, but instead had to clean up the tokens through a manual effort towards the end of the development process. Because this involved finding all occurrences of a token in the code, this was a significant amount of work that was not accounted for.

The actual content assets were generally not translated, but created for a specific target region and language. Therefore, we did not have to provide workflow or a utility for this process. Such requirement would have mandated the use of a stronger tool, for instance Idiom WorldServer.

VIII. JSP TAG LIBRARY FOR FASTER PAGE DEVELOPMENT

Java Server Pages (JSP) technology offers a very powerful feature called Custom Tag Libraries that can improve productivity and maintainability. Tag libraries contain methods that can be invoked through JSP tags, eliminating or at least reducing the need for embedded Java script code (not Javascript!) in a page. We developed a tag library for this project that contained several custom tags used by the Internationalization and Localization features of the site:

translate

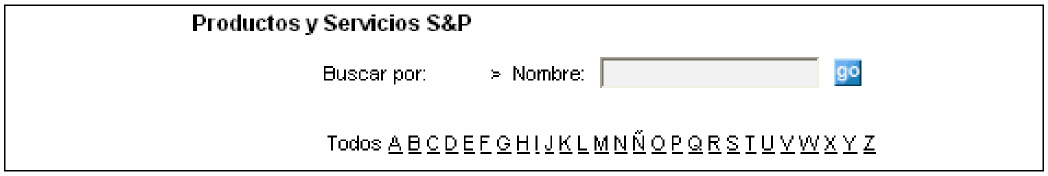
This tag returns a token's translation for a given language and region. Some pages invoke it hundreds of times, therefore its implementation had to be very lean and fast. All translations are kept in memory in a static Hashtable. The Hashtable's *get()* method is used internally to do the lookup. Here is a sample invocation used in a JSP:

```
<sp:translate language="<%=language%>" region="<%=region%>" token="title_s&pequity"/>
```

The region and language parameters are optional as their current values were stored in a cookie and the tag can retrieve them from the Content Server environment if necessary.

alphanav

In some places, the site displays an alphabetical index for quick access to items such as products in a product catalog. Because the alphabet is subject to localization, this custom tag was created that generates and renders this navigational element. Below are examples for the Spanish and Russian language. Note that the Russian alphabetical index includes both the Cyrillic and the English alphabet to accommodate product names in English and Cyrillic.

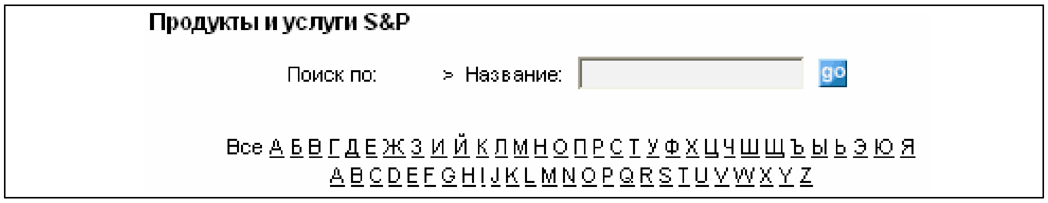


Productos y Servicios S&P

Buscar por: > Nombre: go

Todos A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z

Figure 12 - Spanish Alpha Navigation



Продукты и услуги S&P

Поиск по: > Название: go

Все А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Figure 13 - Russian Alpha Navigation

localdate

Java's on-board date localization and manipulation functions are somewhat limited. To make things worse, we discovered a bug with the Melbourne time zone (GMT+11) in JDK 1.2, which caused TimeZone.getTimeZone() to always calculate Melbourne dates as if it was daylight saving time. This tag was used to format and display localized date and time values and to address the Java bug.

The client's requirements for date formats changed several time during the course of the project and this tag was the single point of control where we were able to quickly change the date presentation for the entire site.

tokenrefresh

This tag was used by the administration interface to refresh the token hash table in memory from the database, to reflect any recent changes and updates.

IX. Character Encoding Considerations

Building a web application entirely based on Unicode proved not to be without complications. The first hurdle consisted of setting up all system components for Unicode support. UTF-8 is the format that has the widest support among vendors and was chosen for this project. Components that had to be configured included:

- Solaris 8 — We used a C Shell profile for the application account and configured the character encoding by setting the following environment variables:
```
setenv LC_CTYPE en_US.UTF-8
setenv LC_NUMERIC en_US.UTF-8
setenv LC_TIME en_US.UTF-8
setenv LC_COLLATE en_US.UTF-8
setenv LC_MONETARY en_US.UTF-8
```
- Oracle 8i — The Oracle installation process configures the database for the character encoding used by the operating system. To override this behavior, a different encoding must be specified during installation. Remember that the size of Oracle character data types (e.g. varchar(32)) does not reflect the number of characters, but the number of bytes. A varchar(32) column can likely not hold 32 characters of UTF-8 data.
- Java — Java internally uses UTF-16 encoding for all strings. On Solaris, the Java runtime refers to the value of the environment variable LC_CTYPE for the transcoding of any IO streams.

The Oracle JDBC driver correctly recognizes the database encoding and delivers data to and from the database in the correct encoding. No additional steps were required to alter the Java encoding behavior.

► iPlanet 6 — The iPlanet application server had to be configured for international use during installation in order to enable UTF-8 support.
For correct Java Server Pages encoding in the servlet container, we used both the relevant JSP page directive and the HTML meta tag on all pages:

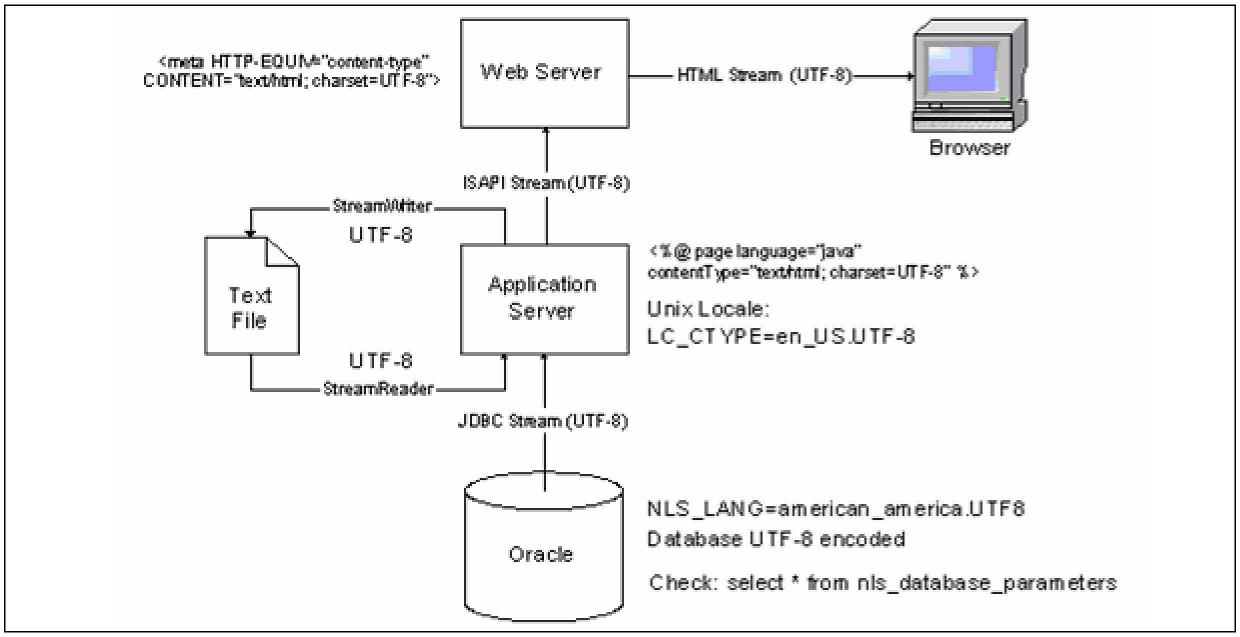
```
<%@ page contentType="text/html; charset=UTF-8" %>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

Components that could not be configured for UTF-8 support included:

► Verity K2 4.0 — A UTF-8 language pack was not yet available for this version of the search engine.

► Pivotal MarketFirst — MarketFirst is an online marketing tool that the client used to manage email marketing campaigns. User registration data were exchanged with this application. Because the version used by the client did not support multi-byte character sets at all, any special characters in user names and address fields were lost during the conversion.

The diagram in Figure 14 illustrates what character encoding steps the data had to pass through as it traveled from the database to the user's browser.



<meta HTTP-EQUIV="content-type" CONTENT="text/html; charset=UTF-8">
Web Server
HTML Stream (UTF-8)
Browser
ISAPI Stream(UTF-8)
StreamWriter
UTF-8
Text File
Application Server
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
Unix Locale: LC_CTYPE=en_US.UTF-8
UTF-8
StreamReader
JDBC Stream (UTF-8)
Oracle
NLS_LANG=american_america.UTF8
Database UTF-8 encoded
Check: select * from nls_database_parameters

Figure 14 - Application data flow and character encoding

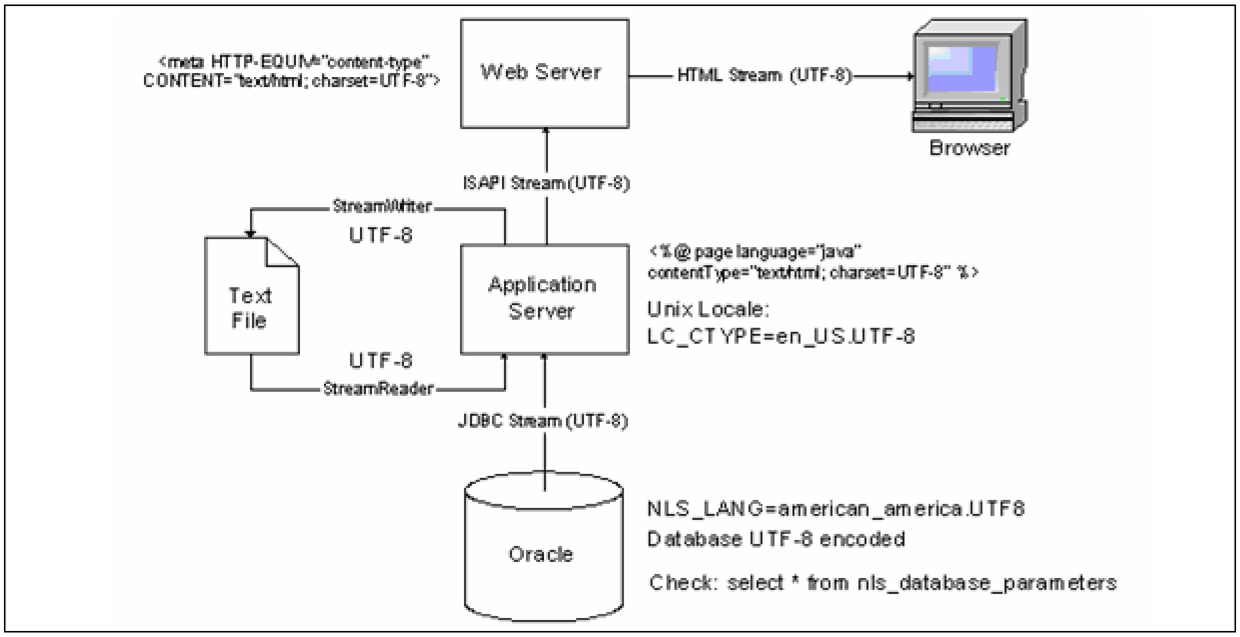
Data were stored in the database in UTF-8 format (provided that Oracle's NLS_LANG parameter is configured correctly). The JDBC driver retrieved data in UTF-8 and transcodes it to UTF-16 for the Java-internal representation, a step that is completely transparent to the developer and is therefore not depicted in the diagram. If the application performed any file i/o, the encoding for the StreamReader and StreamWriter methods had to be specified. Otherwise, Java defaulted to the encoding defined by the environment variable LC_CTYPE, which could be different from what it is intended to be.

The JSP page directive instructs the servlet container to encode the HTTP response with UTF-8. The HTML meta tag instructs the browser to interpret the response stream as a UTF-8 data stream. The data was presented with the correct encoding in the browser.

At some point during the project we experienced character encoding issues that seemed to be arbitrary and only affected certain characters. It turned out that we were working with a development database that did not use UTF-8 and that developers were omitting the JSP page directive to set the encoding. These two factors compensated each other from the perspective of an external observer, but as a result, data were double-encoded in its Java representation, leading to "characters" up to 9 bytes long.

Decoding such a data stream back to its regular UTF-8 representation is not possible without loss and that's why we observed the problems.

Because the search engine used by the site did not provide support for UTF-8 encoded data, it was necessary to separate content into different languages and index each language set into a separate search collection that used its collection's native encoding. We used ISO-8859-15 encoding for all languages except Russian and Japanese. The diagram in Figure 15 shows the process used.



<meta HTTP-EQUIV="content-type" CONTENT="text/html; charset=UTF-8">
Web Server
HTML Stream (UTF-8)
Browser
ISAPI Stream(UTF-8)
StreamWriter
UTF-8
Text File
Application Server
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
Unix Locale:
LC_CTYPE=en_US.UTF-8
UTF-8
StreamReader
JDBC Stream (UTF-8)
Oracle
NLS_LANG=american_america.UTF8
Database UTF-8 encoded
Check: select * from nls_database_parameters

Figure 15 - Search engine data flow and character encoding

A Java program extracted content Meta data from the database and created the language-specific text files required by the Verity collection builder. The text files contained references to the location of any documents in the file system that were associated with the assets. Subsequently, the Verity collection build process indexed the Meta data text files and the associated documents into collections. When the application executed a search through the Java API, the results were returned correctly transcoded to UTF-8.

The following code segment illustrates how the encoding of the output text file for the Collection build process was controlled. A hash table contained encoding names for all allowed languages. The extract object was invoked once for each language and created a separate output file each time. The OutputStreamWriter's constructor accepts a parameter for the target file encoding, which made this code fairly compact.

```
Hashtable encodingHash = new Hashtable();
encodingHash.put("EN", "ISO8859-15");
encodingHash.put("FR", "ISO8859-15");
encodingHash.put("DE", "ISO8859-15");
encodingHash.put("ES", "ISO8859-15");
encodingHash.put("PT", "ISO8859-15");
encodingHash.put("RU", "Cp1251");
encodingHash.put("JP", "SJIS");
String currentEncoding = (String) encodingHash.get(sLanguage);

FileOutputStream fos = new FileOutputStream(bifFile);
BufferedWriter out =
      new BufferedWriter(new OutputStreamWriter(fos, currentEncoding));
```

We used the Verity Java API to perform searches and extract search results from the collections. Verity did not handle the transcoding from the collection's internal encoding to UTF-8 properly unless the VSearch object's selectJavaEncoding() method was invoked.

X. Sorting of Drop-Down Lists

Both the Verity API and Oracle's SQL implementation provide sufficient control for linguistic sorting, i.e. different sorting behavior for different locales. Achieving the correct sorting of text in drop-down lists, however, turned out to cause a significant amount of work.

Imagine a database query that populates a list of country names, for instance. In our case, the query actually retrieved a list of token names and the JSP page replaced the tokens with their translations by calling the *translate* tag for each entry. Because the translations were not known at the time the database query was executed, the sort order of the list could not be controlled. DHTML does not allow changing the sort order of the elements in a list box once it is populated. To solve this issue, it was necessary to first load the list of tokens into a hash table, add the translations, sort the hash table and then use the sorted table to populate the list box. Since this problem surfaced late during user acceptance testing and it had to be fixed for every list box on the site, a lot of extra code was written.

In the project that I am currently working on, I decided to give up the principle that all translations are retrieved from one database table through the *translate* tag. Instead, all read-only reference values are stored with all translations so that one database query returns the values for the actual target language in the correct sort order. While now there are translations in multiple tables in the database, this approach helps avoiding any issues with sorting.

XI. Summary

International content can be characterized by four attributes: region of origin, destination region, regions covered by its content, and language. Such content can be targeted to certain regions on a Web site in two ways: either by direct placement or through keyword tags. Both methods have their merits and can be used together.

Unicode implementations are still the less traveled road for many software vendors. Deploying and customizing products for interoperability in a Java environment with UTF-8 encoded data still causes a surprisingly large number of pain points that is difficult to plan for if the limitations of these products are not known ahead of time – which is rarely the case.

Despite the obstacles, this project was a success that enabled the client to streamline their Web publishing process worldwide so that the content management can be managed centrally by 1.5 technical resources. The Web site won the 2004 award for Best Institutional Services Website by the Web Marketing Association.

XII. Biography

Christian Donner has been building software since 1985 – initially in Austria, now in the US. Over the course of his career, he rarely did the same thing twice. Consequently, his portfolio includes such diverse projects as an insurance policy management application, an accounting and billing system, a data warehouse, middleware infrastructure, and several multi-media solutions.. He is equally experienced with native client applications running on Unix, OS/2, and Windows, and web clients. He has 4 years of experience implementing and customizing Fatwire Content Server.

His professional creed has remained the same throughout his career: *to solve a business problem, apply common sense before applying technology*.

Christian holds a masters degree in electrical engineering from Vienna University of Technology and is a Sun Certified Enterprise Architect for Java. He is currently a senior consultant at Molecular, a Massachusetts technology consulting firm, and lives south of Boston with his wife and his two children. He can be contacted at pubs2005@cdonner.com.

References

[1] *Oracle Globalization Support* (http://www.oracle.com/technology/tech/globalization/index.html)
[2] *New Datetime Data Types in Oracle9i* (http://www.oracle.com/technology/products/oracle9i/daily/may02.html)
[3] Mark Davis, *Forms of Unicode* (http://www-106.ibm.com/developerworks/library/utfencodingforms)
[4] *HP Internationalization Checklist* (http://h21007.www2.hp.com/dspp/tech/tech_TechDocumentDetailPage_IDX/1,1701,3622,00.html)
[5] *IBM Software Internationalization Guide* (http://oss.software.ibm.com/icu/userguide/i18n.html)
[6] *Sun Microsystems Java Internationalization Resources* (http://java.sun.com/j2se/corejava/intl/index.jsp)