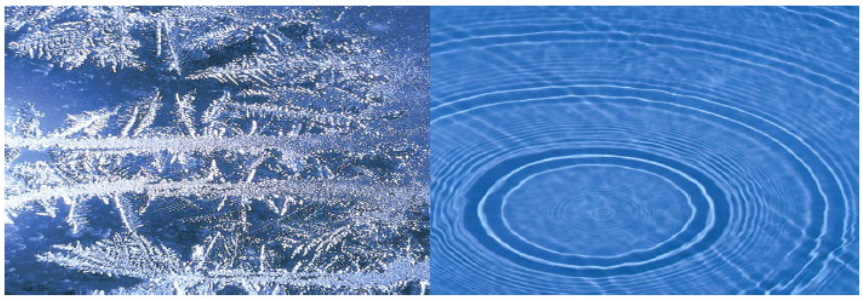


VIOMA

[easy content management]

## VIOMA Content Management White Paper

Welcome to the wonderful world of content management.

This document serves several purposes:

Chapter I provides a background to content management, together with a brief sketch of the problems facing those companies that aim to extend their presence on the Internet.

Chapter 2 is an explanation of a concept that has become increasingly unclear with increased usage, the concept of content management. This chapter also briefly explains the principles and methods of this concept.

Chapter 3 can be regarded as a guide for presumptive buyers of content managers, with practical advice on what to look for and what should be the minimum requirements for a complete system.

Chapter 4 is a presentation of VIOMA Content Management Server ™ 4.0, Content Solutions, Inc.'s own Content Manager. Hopefully, the preceding chapters will have given enough knowledge on the subject of content management to make the advantages of both technology and method obvious

Chapter 5 is a description of the most important elements in a content manager, in particular those XML-based technologies that make modern content management possible, and the way in which they work. The aim of this chapter is not to confuse the reader, but to convey a deeper understanding for the inherent possibilities in a content manager, an understanding that is difficult to reach unless there is a fundamental understanding of how a content manager works.

At the end there is an appendix of XML examples and an explanation of some of the terminology used in this document.

**Table of Contents**

# Chapter 1: The Content Management Challenge

In just a couple of years, the Internet has grown from a kind of digital playground to a mature business platform. The possibilities have never been greater; neither have the challenges. The volume of company websites doubles every year. At the same time, new functionality is added, such as audio and video, as well as sophisticated profiling functions to customize the content for the benefit of the user. Added to this is the constant development of Internet-enabled hardware.

All this creates a need for tactical as well as strategic decisions by the companies that aim to make the most of the possibilities on the Internet.

These decisions will not come easier the longer they are deferred. But still many companies continue to create and administer their websites with the same ad hoc methods and homemade tools as earlier. The result is overburdened staff, costly routines, and an outdated website that fails to meet customer expectations.

In spite of this, there is reluctance to implement disciplined methods and professional tools. Perhaps things have developed too quickly. After all, not long ago a website could be administered by a few college students in a basement somewhere. But today, a website is a business channel of great importance and deserves both the time and resources that have been spent on other more traditional channels.

Those working with IT are well aware of the challenges that lie ahead, but those that are not so well acquainted with the problems may need a more in-depth description.

## *Information gridlock*

The greatest difference between the Internet and more traditional business channels is that the Internet is far more complex than anything else we have ever experienced. The Internet demands more from both the sender and the receiver, but it is naturally in the sender's interest that everything works without jarring anywhere. To begin with, he must satisfy the demands of those that create content, from web designers with comprehensive knowledge of HTML, to market directors, whose competence may not rank very high when it comes to technological issues. But satisfying the demands of those that create content is only the first step. The website must also be able to handle content for various types of users and various types of media. All this means vast quantities of information that must be structured in some way to reach the correct user in the correct format. Unfortunately, the result is often information gridlock, better known as Webmaster bottleneck. This occurs when the amount of information exceeds the ability to handle it. Webmaster bottleneck is, however, a misnomer, since this problem is not to be blamed on the Webmaster. No one can handle challenges like this without appropriate tools. But regardless of whose fault this may be, the result is the same: Time and money wasted, and customers that take their business to competitors with better systems.

## *Improvised routines make improvised websites*

Marketing directors can often be heard lamenting the lack of a coherent look and feel on the company website. Many websites are the unfortunate result of documents from different departments, different tools, different guidelines, and different style sheets. The consequence is a website that looks as if it were created not by one, but by several different companies. Adding to the confusion is the fact that different navigation

methods are used on different areas of the website, which confounds the user and leads him to abandon the website in search of one which is more user friendly. A poorly constructed and maintained website may do double damage by not only failing to serve its purpose, but also by counteracting marketing efforts in other areas.

## *Customers expect individual information*

The Internet users have come to expect dynamically assembled information delivered in real time. The ubiquitous news sites are a good example, as are sites aimed at the financial markets or weather sites. On a more practical level, it would be possible to track deliveries or supply levels on the Internet. To achieve this, the content on the website must be structured, stored and constructed so that it can be delivered dynamically, or customized for a specific customer at a specific time. In other words, the website must include not only content, but also functions that allow the content to be structured according to the needs of the user.

## *Diversity of clients raises demands*

The world of information technology is rapidly expanding from a PC-centric world to a more diversified world where users seek information from a variety of clients, such as palmtops, TV sets, and mobile phones.

Consumers, as well as producers, have realized that most users do not require the full capacity of a PC, but that they in many cases prefer thin services such as E-mail, personal information services such as calendars, and access to the Internet.

Obviously, this raises the demands on those producing the information. It must be made available in several different ways via several different media, and it goes without saying, that this is yet another burden on an already overburdened website administrator.

## *Traditional websites will soon be history*

According to the Forrester report "Managing Web Content", those companies that fail to make the transition from manually built pages to elements stored in a database will store their web content in a format that is soon to be abandoned. This means that the value of the data drops to almost zero and the value of the online service goes the same way. Naturally, this restructuring is not an easy task, but those companies that start early will have a considerable competitive advantage over their more laggard competitors.

## *XML means new challenges*

XML is the latest web standard for creating documents and data on the net. The major difference from HTML is that, with XML, it is possible to tag information semantically, or to put it a little differently, the tags also carry information that describes the purpose of the information. This is a fundamental distinction. This is what makes it possible to save all information in one place, while it is still possible to deliver it via different media.

By fragmenting information into independent parts, it becomes possible to handle it in ways that were previously impossible. For instance, it becomes possible to adapt it to different types of hardware, it is easier to customize for different users, and it is far simpler to store in a database, which altogether is a good

foundation for the next emerging market on the Internet: business-to-business.

## *Conclusion*

The Internet has developed at an incredible speed over the last few years and there is nothing saying that it will slow down within the foreseeable future. Today, the management of information is largely based on dated, obsolete, or insufficient technology, which means that we have reached the limit of what it is possible to achieve.

XML is a new standard for information management. By dividing the information into smaller parts and tagging it semantically, we achieve a flexibility and structural consistency which simplifies adjustment to different types of hardware as well as user needs. This in turn makes information far easier to handle and administer which means firstly that larger amounts of information can be handled and secondly that a larger number of users can be reached.

# Chapter 2: What is Content Management?

Content is all the information that a company needs to operate. It could be anything from a customer register to the annual report or the business logic. Thus, content is not only the information on the website even if in the Internet-focused world of today it has more or less come to acquire that meaning. From this follows that the term content management should encompass all types of media and channels, such as fax, WAP (Wireless Application Protocol), TV and palmtops; that is, all the channels that a company wants to employ.

At most companies, there is a realization to the need of an effective means of structuring and delivering information, but most companies do not know where to begin. The first thing to do is to agree on what content management really is and how it works. This chapter will give an overview of content management, followed by a more detailed description of the different parts of the process.

## *Content Management - an overview*

Content management need neither be complicated, nor difficult. In fact, it should be the direct opposite. The different steps of content management are actually quite fundamental, regardless of the tool chosen:

1. Professional web designers and developers structure content by creating input forms and templates.
2. Content authors then use these forms and templates. There are several different methods to choose from, but, regardless of whether you prefer a word processor, an input form, or some kind of developer's tool for XML, the end result is the same: Content is structured into XML format and saved to a database.
3. The information is stored as fragments in a database, from which a publishing engine extracts it and applies the templates to structure and customize it for different types of hardware and users.

This was a very simple sketch of content management, but with this in mind we can study the different parts of the process more in detail.

## *Structuring content*

The first step is always the most difficult. In this case the first step is a decision to structure the content of the website and store it in a database in order to create information that:

- Is easy to access, easy to share and easy to update.
- Can be assembled dynamically to meet different needs.
- Has a consistent look and feel throughout the website.

## *Methods for structuring content*

There are several different methods to choose from when it comes to structuring information. The choice should be based on the nature and purpose of the information.

- Forms. Web developers and -designers can create input forms that content authors can use for creating XML-based documents.
- Templates. Just as input forms define the structure of what is put into the database, templates are used to structure and format the information that is retrieved from it.

- XML editors. Professional developers can structure information by using an XML editor. In this case the developer creates structured documents according to a predefined schema and saves the information as metadata directly to the database.

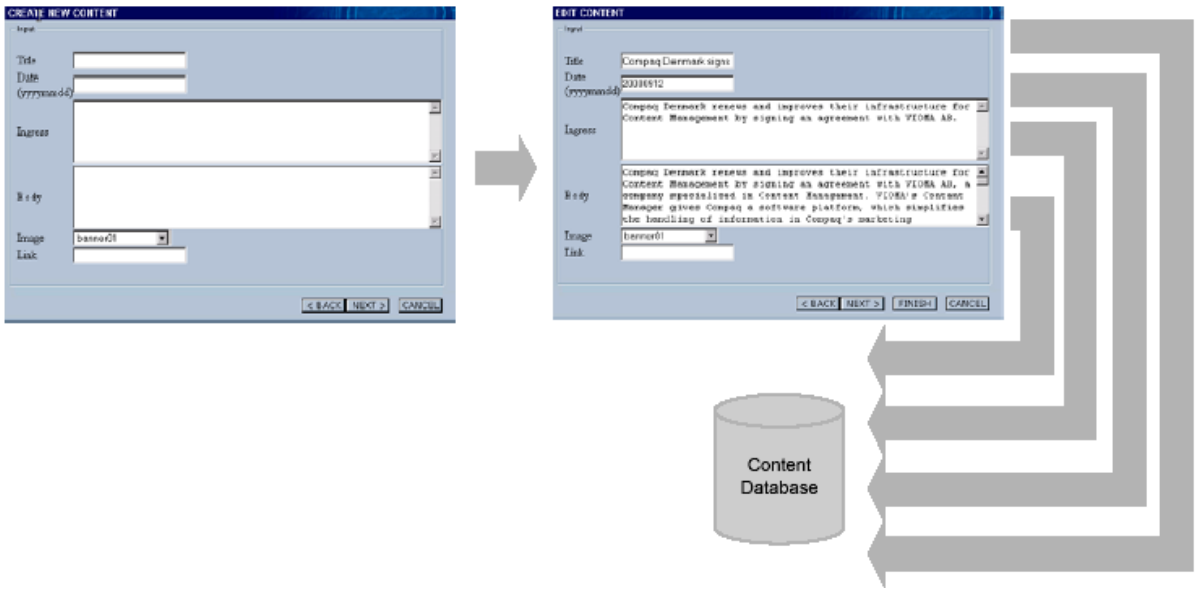
## *Structuring content with forms*

This is one of the most efficient ways of structuring information. These forms are usually HTML forms created by web developers. You simply fill in the fields and click the save-button to save structured information to the database. When the data is processed according to the instructions for the form, it is automatically turned into intelligent XML fragments and saved to the database. An example: To structure training materials for an organization, the web developer could construct a form that contains a Topic field, a Subtopic field and a Module content field. To create new training materials, all the author has to do is to open the form "Training Module", fill in the form fields, and save the information to the database.

By using forms, traditional documents can be broken down into fragments with simple references that can be assembled in different ways to meet different needs. This in turn means that one set of content can be used for different purposes, which raises the value of the information.

## *Structuring content with templates*

In the traditional world of word processing, templates are used when creating a new document. These templates are used to provide a visual consistency between different documents created by different authors. Each document is then stored as a specific item even though the differences between them may be only very slight. With a content manager, the information is stored as raw, unformatted XML-based data. A publishing engine then uses templates to transform this data to data that suits the hardware or user that requests it. The information as such is stored only once and in one location only, which means huge savings in both administration and use of resources. The publishing engine performs three main tasks: it extracts the information from the database, transforms it into code that suits the hardware or target group in question, and formats this code to ensure a consistent look and feel of the information.




*Information that is stored as XML metadata in a content database.*

## *Structuring content with an XML editor*

XML editors are often used to convert existing static web pages into XML-based data. The easiest way to do this is to hire a website developer who uses a tool such as SoftQuads XMetaL, DTDs, or Schemas to save the information to a central database.

## *Creating content*

There are several different methods to let content authors create content. One of them is to use Microsoft Word together with a database link that saves the content directly to a database. Another method would be to work in an integrated environment with a text editor and forms that save the content in the database. A third method is. to work with an XML editor linked to the database. Every method has its own specific advantages and the best content managers have the flexibility to accommodate all three methods. The most important thing is user friendliness. The best content authors are not necessarily the ones with the most technical knowledge, and they should be spared the technical aspects of their task, in order to let them concentrate on what they do best.

Making the website more accessible has several advantages:

- The website becomes more updated, since content creators quickly and easily can create and revise their own documents.
- Publication costs decrease, since the process becomes independent of costly technical expertise.
- Information reaches the market more quickly since the number of steps between creation and publication is reduced.

This does not mean that everyone can publish everything on the company website. Most content managers are equipped with variable security settings which means that a website administrator has to approve the content before it is published on the website. He is, however, relieved of the time- and money-consuming transformation, and content authors are relieved of a long wait before they can see the result of their efforts.

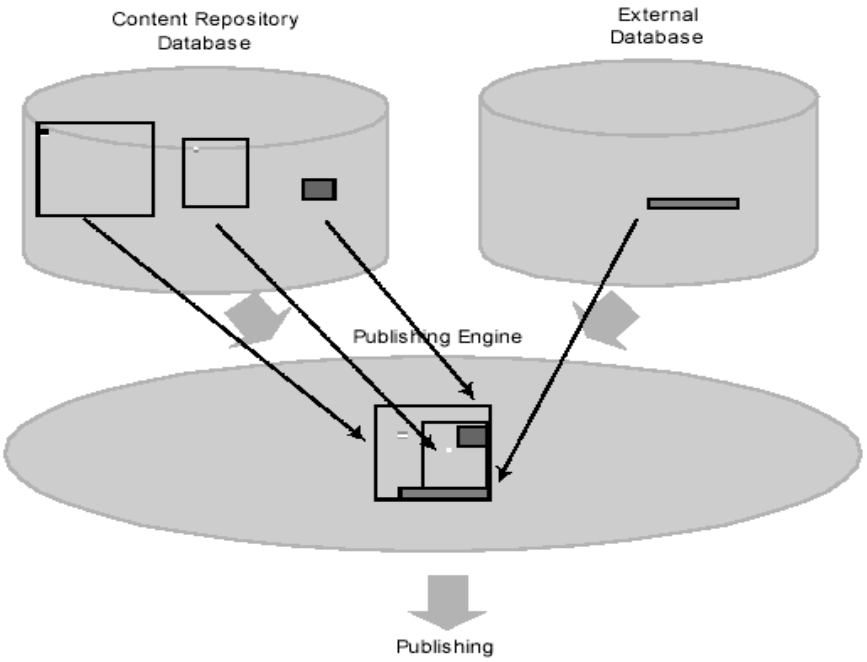
## *Managing and storing information*

Most content managers are built on a database where content is stored. There is no need for technicalities at this point, but storing the company information in this way offers several advantages:

- The information is stored in one location, which makes it easy to administer and update.
- If the information is stored as XML metadata, it can be assembled and formatted in several different ways, although only one version of the source material is needed. This eliminates the risk of double storage of the same data and also of the risk that different departments use different versions of the same source material.
- If content is stored as XML metadata, all relevant information can be extracted from the database, which leads not only to increased precision but also raises the value of the information since it becomes more versatile.

## *Publishing content*

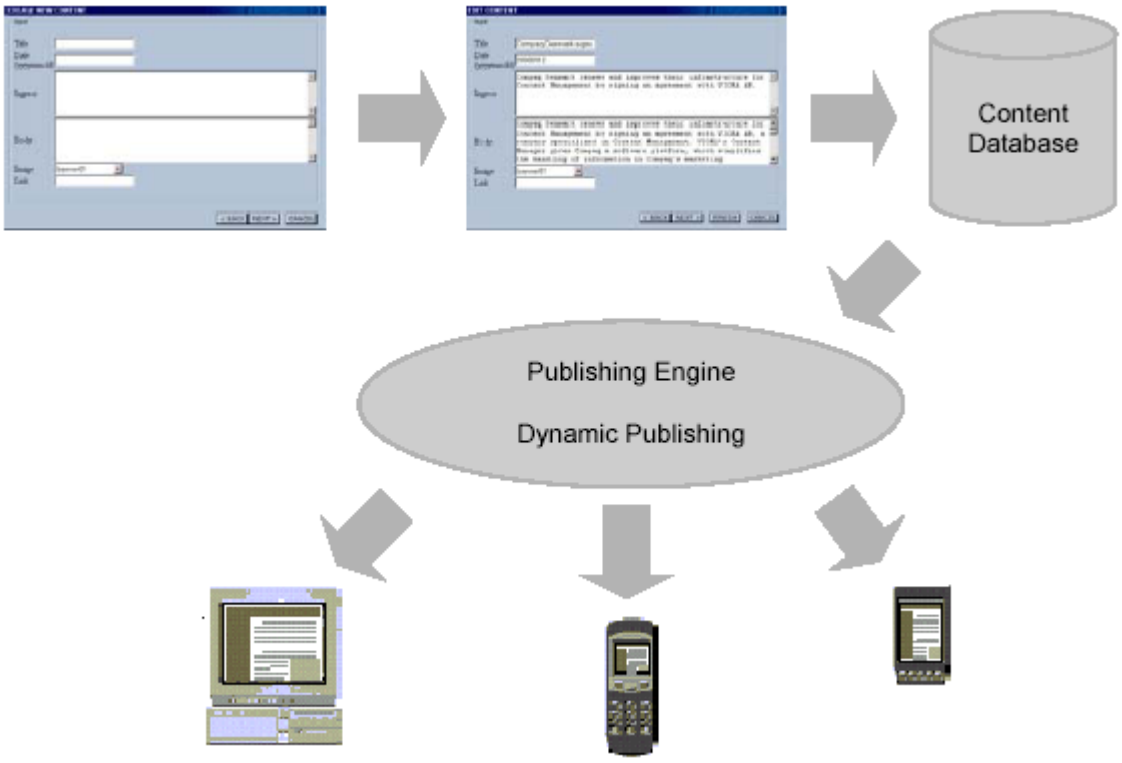
If the database is an information storehouse, then the publication engine is the storage worker. It is the publication engine that receives the orders, collects the articles, and delivers them according to the specification. The publishing engine is a kind of link between the database and a web server. But it is more than just a transmitter; it also performs functions to further refine the information.




The publishing engine enables dynamic publishing for different needs.

- A built-in flexibility makes it possible to construct documents dynamically, which means that the user always has the latest version.
- The ability to recognize the identity and hardware of the receiver, regardless of whether it is a retailer with a WAP telephone or a field technician with a laptop.
- The possibility to work from different sources, including document fragments, external databases and business logic, which means that the user will have an optimal result from all available sources.

Naturally, the publishing engine also supports various customer profiling functions. When these are connected to the functions for business logic, the result is a web page that is constructed completely according to the requirements of both the company and the customer.




*The content management process.*

## *Conclusion*

Content management is a concept that is both vast and wide, and this is only a brief sketch of the technology and the advantages of XML-based content management. The technology is based on XML, a computer language that breaks down information into fragments, which are stored in a database, from which a publishing engine can extract and assemble them.

The great benefit is simple information management. There is no need to code text and other material that is to be published on the web, which means increased accessibility for content authors and a more updated website.

This system also demands less resources than most traditional systems since the information is stored only once, in one location. At the same time, it is more flexible, since the XML format makes all the information equally accessible to the publishing engine, which can retrieve all relevant information, format it according to the needs of the user, and deliver it.

# Chapter 3: To evaluate a content manager

Almost all content managers offer powerful publishing engines and access to company databases, but if you want a complete product and to get maximum use of all the advantages in a content manager, there are certain functions to look for when it comes to evaluating the product. This chapter will discuss these functions and also offer some handy hints as to which are most important for successful content management.

## *Easily deployed and scaled*

A content management system should, ideally, be easy to install and scale. Web-based interfaces are generally to be preferred since they do not demand any desktop installation. Web-based interfaces are also easily scalable, since new users are added simply by directing them to the correct URL and letting them log on to the system.

## *Centralized administration*

A good content management system centralizes access to the systems that approve, update, and deliver information. The very best systems can also boast variable security levels and uncomplicated access to documents, templates, and configurations.

## *Easy to structure and format content*

Designers and developers want a simple method to control the look and feel of the content. Therefore, the system must have straightforward routines for creating company templates and for the application of these in varying situations and environments.

## *Separate content from layout*

When evaluating content management systems, make sure that the system supports a separation of content from format. As was shown in the previous chapter, the company data becomes much more flexible this way and can be assembled and formatted in a variety of ways to suit different kinds of hardware or customers.

## *Separate content from business logic*

When evaluating content management systems, make sure that the system is open, with interfaces that allow the development of business logic, separately from content. This functionality can be developed by the company or in the form of business systems.

## *Support for popular content-creation tools*

The ideal content management system does not tie the company to a single proprietary tool, but provides the flexibility to use the tools that best address the company needs. A good content manager should support a wide range of the tools used to create XML and XSLT-based templates, as well as those tools that structure and contribute information to a central database. This means that non-technical staff can use a

simple word processor to create content, while a web developer has the option of a professional XML editor.

## *Easy to create, find, and share content*

It should be easy for users to create and share information without mastering the niceties of HTML. Ideally, they should be able to use the tools they already know, such as a simple word processor that demands no knowledge of HTML. In addition to this, the best content managers give non-technical authors access to a content repository, where they can share, contribute, or store information without any knowledge of data structures, complex formatting rules, or HTML code.

## *Support for Version Control and Rollback*

A content management system should keep track of changes made to documents by maintaining unique revisions for each change. The user should be able to restore content to previous versions of a document whenever necessary. Also, the system should be able to provide an audit trail detailing changes made to content - what, when, and by whom.

## *Powerful centralized database*

All content managers must have a powerful, centrally administered, database, where all information is stored and retrieved. Like most systems, it must be scalable to cover the entire organization, sufficiently robust to handle several websites, and at the same time be able to handle simultaneous hits. In addition to this, the database must support open XML standards and work with standard platforms such as Windows. The best content managers also work with industry-standard relational databases such as IBM's DB2, Oracle's Oracle 7 or later, MySql, and Microsoft's SQL Server 7.0 or later.

## *Powerful publishing engine*

Any content manager worth considering must have a powerful publishing engine that can localize separate document fragments, assemble them dynamically, and then format them for the target audience or device so that the information becomes both custom- and media-oriented. The publishing engine should also provide interfaces to business logic for customer profiling and relationship marketing.

## *Compatibility with open standards*

If your company plans to install an enterprise-wide content management system, it is important to select a system that supports open standards, especially XML. When evaluating a content management system, make sure that it integrates seamlessly into your environment, and that in addition to XML, it supports HTML, XSLT, WML, Java, and JDBC- and ODBC-compatible external databases.

## *Conclusion*

When evaluating a content manager, look for easy deployment, easy scalability, centralized administration, and compatibility with open standards. The most important thing to look for is, however, ease of use. After all, the whole point of content management is to make life simpler, not only for site administrators, but also for content authors and web developers.

# Chapter 4: VIOMA Content Management Server ™ 4.0

As has been shown in the previous chapters, a good content manager should offer functions that, even if they are not strictly necessary, offer the added value that makes it possible to draw full advantage of the product. It goes without saying that a complicated system with labyrinthine routines will not gain the approval of the user, no matter how powerful or useful it is.

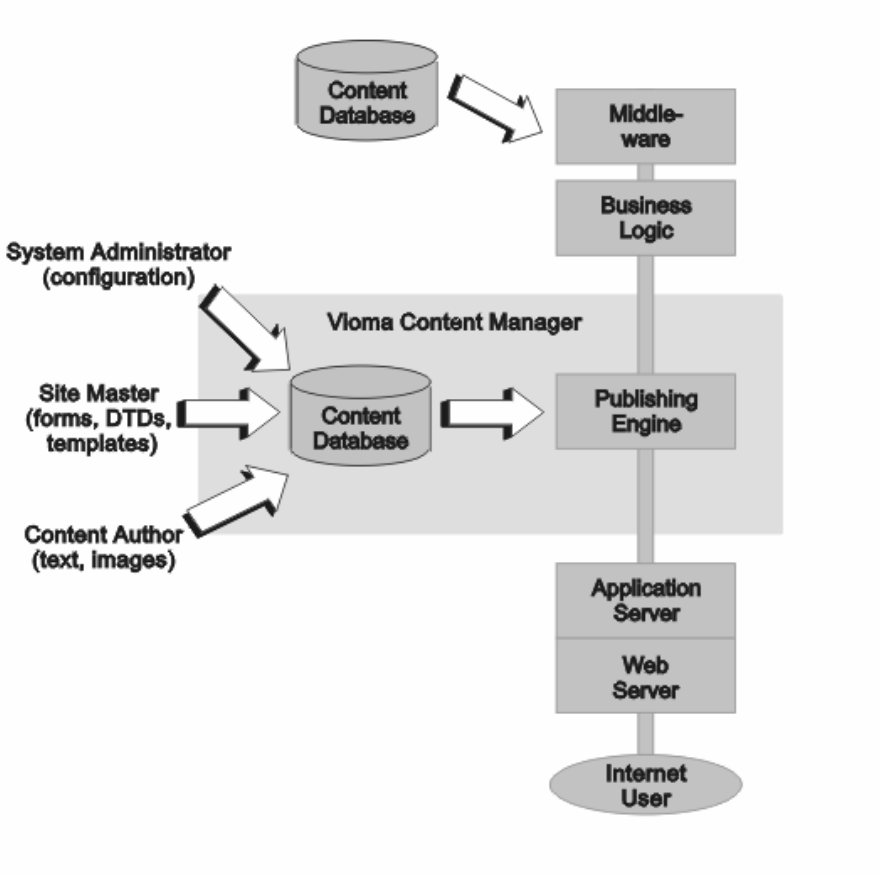
VIOMA Content Management Server ™ 4.0 demonstrates our commitment to easy content management. This product offers an elegant yet intuitive user interface, with three distinct environments that have been carefully crafted to the needs of the user.

VIOMA divides setup and maintenance of a web site into three areas, each one managed by a different user category:

- **The system administrator**
  The system administrator sets up hardware, software, and communications. The system administrator is also responsible for the integration of VIOMA with surrounding applications and devices. System administration work is carried out in the System Administrator module.
- **The site master**
  The site master sets up and creates forms, DTDs, templates, user accounts, workflow, and stages. This work is carried out in the Site Master module.
- **The content author**
  Content authors enter and maintain the information published on the site. Certain users may also be given control over what is published, by means of approving information using the built-in workflow control function. Also certain users may be given the capability to add, delete, and rename containers that hold content, a function normally reserved for the site master. Authoring, workflow control, and version control and rollback is carried out in the Content Author module.

## *System Architecture*

VIOMA Content Management Server ™ 4.0 consists of the content repository, a publishing engine, and a web interface for entering and editing content.




*Overview of a VIOMA installation*

VIOMA can be integrated with custom databases and business logic to form an integrated publishing environment, with all publishing managed by the publishing engine.

This chapter is a brief overview of the features and benefits of the three components of VIOMA's interface. You will notice that a number of functions are described several times. This is done in order to describe the various features as comprehensively as possible. Once you have an understanding of how the product looks to the user, it will be described in more detail, in order to give you an appreciation of the powerful yet flexible features offered.

## *Easy to configure, deploy and administer*

VIOMA Content Management Server ™ 4.0 provides a web-based System Administration environment that simplifies the process of managing website assets and users. This environment offers central control of functions like configuration, hyperlinks, security, templates, and content. There are content managers that require a team of experts to get the software up and running. VIOMA has been developed to fit seamlessly into your enterprise without a long and complicated setup process. The automated setup program will guide you through the installation and configuration. VIOMA can be quickly and seamlessly installed by using security and administrative functions already familiar to your system administrators. VIOMA is also easily configured to work in concert with a variety of popular databases.

## *Flexible and scalable deployment*

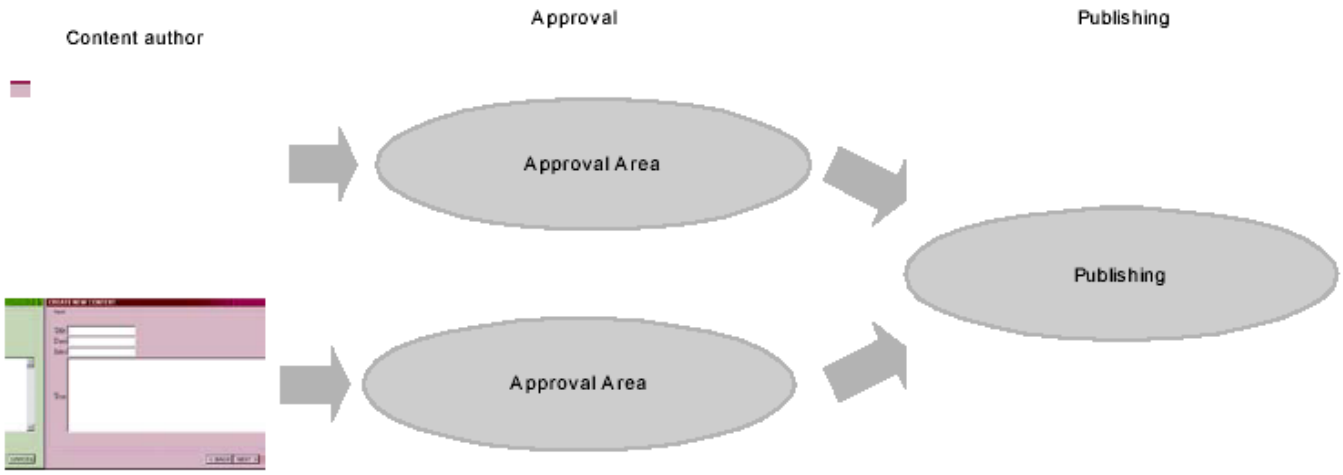
VIOMA can be easily deployed and scaled since there is no desktop installation required to add new users to the system. With VIOMA, the site administration simply assigns the necessary security to add the user or web designer to the system and then sends the appropriate URL to the new user to inform them of the location of the template authoring or content authoring environment. Equally important, VIOMA can utilize multiple processors on SMP (Symmetrical Multi-Processing) computers, which means it can be easily scaled for industrial, heavily trafficked websites without suffering performance degradation.

## *Flexible security*

VIOMA has an advanced security system that integrates with open standards, such as webserver authentication and LDAP (Lightweight Directory Access Protocol). That means that site administrators can provide a variety of security levels for site administrators, template authors, and content authors using built-in security already provided by Windows. Equally important, VIOMA uses a security model where accessing protocol controls authentication, which means that Client Certification technology can be used for web browser authentication or the SIM-ID for WAP authentication.

## *Structured workfiow and document approval*

Without a structured systematic approach, document creation, collaboration, approval, and ultimately publishing can be, at best, a challenging task. But VIOMA streamlines workfiow by dividing tasks and security levels among various groups within the enterprise. The effect is that VIOMA creates a division of labor, giving content authors the ability to create and update content, template developers the ability to structure and format content, and the site owner the tools to approve and publish web content. VIOMA provides an automated staging area for content by flagging it when it is stored in the content repository. Since security levels control access to the repository, only authorized personnel have the necessary permission to publish content on the web.




*Workflow approval.*

## Versioning and rollback capability

VIOMA Content Management Server ™ 4.0 automatically maintains source code control for every document created in VIOMA, content as well as templates, forms, DTDs, etc. Each time a document is changed, a new revision of the document is created and stored in the database. Content authors can revert to a previous version of a document by requesting the system to do a Restore File; he is then presented with a list of previous revisions of the document, from which he can choose. Also, site managers can revert all content back to a previous date and time.

### *Easy site management*

With VIOMA, website assets such as configuration settings, site mappings, templates, and content are centrally stored in the content repository database. With VIOMA's web-based administration environment, site owners can easily find and manage website assets, which means reduced IS maintenance hours, faster publishing times, and reduced publishing costs.

### *Easy data structuring and transformation*

VIOMA provides a built-in template authoring environment for creating content input forms and for creating XSLT and CML templates that transform and format XML-based content. This environment is seamlessly tied to the content repository so that template authors can create templates and save them in the database by a simple click on the save button. In addition, this environment offers a variety of other benefits described in the following sections.

### *Centralized template authoring environment*

In traditional web environment, locating style sheets and updating them is not always an easy task, since they can be scattered over different servers and folders. They can also be integrated into the document itself, which makes them even more elusive. But in VIOMA, both the XSLT and the CML templates are stored in the content repository, which makes them easy to locate in the Site Master environment. Moreover,

VIOMA offers enterprise-wide security, so that only users authorized to use the Site Master environment have permission to create, update, and store the templates that define the look and feel of web documents.

## *Centralized look and feel*

In many organizations, content is formatted with a variety of style sheets, each developed independently by separate departments. The result is often a website closely resembling an office party at the UN General Assembly: i.e. an eloquent lack of both visual and functional coherence. VIOMA solves this problem by security controls in the site administrator's environment, which means that only authorized web developers and designers can create formatting templates. Once the templates are created and saved to the content repository, the site administrator maps them to the proper folder. Centralized control saves not only time and IT maintenance, but also ensures that the corporate website has a consistent look and feel across the enterprise.

## *Support for content input forms, DTDs, and schemas for structuring user data*

The Site Master environment provides support for the easy creation of XML-based data input forms, DTDs, and schemas that essentially model data for your enterprise. With input forms, content authors can create pre-structured content that conforms to organizational standards, without worrying about formatting or document structure. In addition to this, in the website administrator's environment, developers can create DTDs and schemas to define the XML tags used to create and structure content across the enterprise, and between businesses with disparate servers and database systems.

## *Support for open standards*

VIOMA Content Management Server ™ 4.0 provides support for existing and emerging Internet standards, so that templates can be created using HTML, CSS, XSLT and VIOMA's own Content Markup Language (CML).

## *Easy template auditing via HTML and CML*

VIOMA simplifies the creation of templates with its Content Markup Language (CML). By using CML in combination with HTML and CSS, template authors can quickly and easily create powerful templates that not only format content but also query the content repository to create dynamic web documents in real time. CML lets developers create templates that transform XML source content stored in a database into a wide variety of output media, including HTML for the web, WML for WAP, and PDF for print.

## *Easy content authoring*

The VIOMA Content Authoring environment offers a built-in web-based word processor that enables non-technical authors to create and save documents to a content repository. Moreover, the content authors can easily find and update documents using the file management system in this environment. VIOMA does all the work behind the scenes, letting content authors concentrate on their work without having to deal with the complexities of creating XML-based data, or saving it to a database.

## *Separation of content from format*

In the Content Authoring environment, content creators can create documents in a word processor or an input form without having to struggle with complex formatting in HTML, XML, or CSS code. By empowering non-technical authors to create content for the web and update it themselves, content can be created at a fraction of the cost associated with having web professionals translate documents into HTML code. The result is not only significant savings in time and money, but also a more responsive website.

## *Easy database storage of XML-based content*

VIOMA Content Management Server ™ 4.0 provides a data access interface for storing XML content, XSLT/CML templates, and system configurations in industry-standard database systems, including IBM DB2 version 6.1, Oracle 8.0, MySql, and Microsoft SQL Server 7.0. The content manager's web-based environments are integrated with the database through the Systems Services and Data Access Components of VIOMA Content Management Server ™. Once VIOMA is set up and configured for the appropriate database, users of VIOMA can create, save and update content using the file managers provided in the environment. In short, all content across the enterprise can be seamlessly stored in a centralized database without exposing end users to all the complications of a backend database.

## *XML-based storage of text, image, and binary objects*

VIOMA implements hierarchical object storage so that the tree structure is preserved within the database. This means that the content can be finely granularized and thus prepared for a wide variety of purposes, including output for WAP, PDF, interactive TV, or traditional web display. By using IBM DB2, Oracle, MySql, or Microsoft SQL Server for underlying storage, VIOMA supports storage of text, images, and binary objects. What is more, VIOMA tracks meta-information on all content objects in the content repository database, enabling the meta-information to be used for information matching, searching, and profiling.

## *Maintenance*

The advantage of working with a database is that it allows your organization to streamline processes while maintaining a consistent and easily manipulated set of content. This means that your organization will spend less time on maintenance of the website and more time on actually creating new content.

## *Reduction in data redundancy*

Storing XML-based content in a centralized database reduces content redundancy, since the same content can be used in a variety of ways. In traditional content storage, every document is stored separately, even if it is only an address that differs among them. But by storing content as XML data, the same source document can be transformed dynamically and formatted in real time to meet the needs of a certain target group or device. This means a dramatic reduction in the amount of stored data and of the cost associated with this.

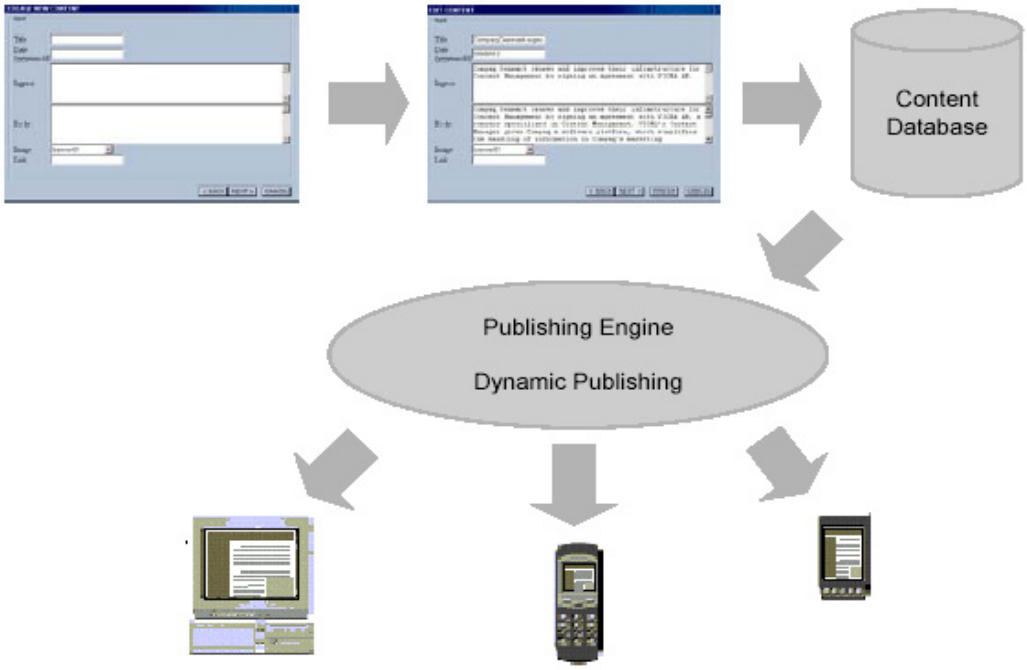
## *Dynamic publishing and formatting with XSL/CML templates*

In many cases, documents are created as a response to a customer query, which could be something as basic as a request for a web page. Since VIOMA stores information as XML metadata in a database,

content fragments can be extracted from this or external databases to dynamically construct a document based on a customer request or the business logic applied to the document.

VIOMA Content Management Sever ™ 4.0 provides a powerful publishing engine for dynamic delivery of customer-aware and device-aware information, so that your customers get web-based information that is both targeted and timely. The publishing engine is actually a Java servlet that offers great flexibility for applying business rules for customer profiling and for accessing data from external databases. In addition to this, the publishing engine supplied with VIOMA comes with built-in business rules for device detection so that templates can be applied to documents to dynamically format them for different types of clients, such as PC, PDA, TV, or mobile phone.

One of the primary advantages that Vioma has over simpler systems is its support for the execution of custom business logic to process Web pages. Development and processing of forms is generally one of the more difficult parts of website development. By using Vioma's libraries and your own business logic, you can develop a truly database driven website, where HTML is dynamically generated from database fields, user input verified and processed, and user answers stored back in the database. See Appendix 2 for more information about how Vioma uses the FormMap functionality of the Barracuda project

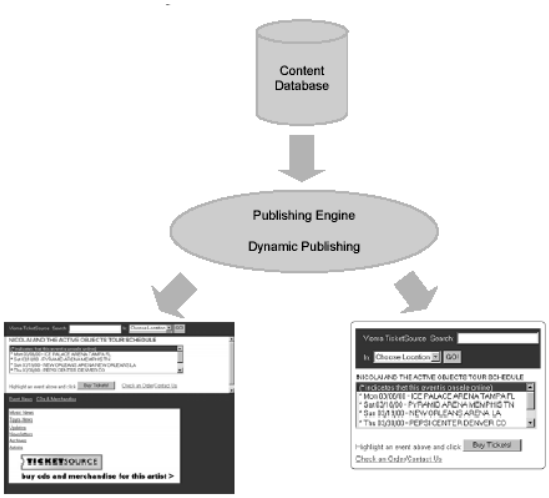



*Dynamic publishing and formatting with templates.*

## *Dynamic formatting with templates*

VIOMA's publishing engine dynamically applies CML- and XSL-based templates for intelligent and consistent formatting of web documents across the enterprise. Templates are created by web developers and designers and mapped to nodes by the Website administrator. These templates are then automatically applied to the content to tailor it for a certain target group or device. Storing transformation and formatting information in templates, rather than in the document itself, gives increased flexibility and control over

website look and feel. Perhaps equally important is that the separation of formatting instructions from content dramatically reduces the amount of maintenance required.




Dynamic publishing to different channels. Same content is adopted to different channels with templates.

## *Flexible connectivity*

VIOMA provides a wide variety of connectors to extend the breadth and reach of your company's communications. The connectors include:

- A Java sevlet-connector to allow publishing of web content via a web server that supports servlets, so that content can be delivered using an HTTP-protocol.
- A Nokia WAP Gateway Connector, so content can be delivered to Internet-enabled mobile phones using a WAP protocol.
- An RMI/CORBA Connector, enabling VIOMA to be integrated with enterprise solutions and components. Perhaps more important, however, the RMI/CORBA Connector opens the door for powerful business-to-business communication by using XML as the format for data exchange between disparate business systems.
- A TCP/IP Connector to store and receive content, and for easy integration with other third party tools and solutions.

## *Recycling*

VIOMA's publishing engine leverages productivity since the information - once it has been created - can be used for a wide variety of purposes. For example, a product description can be authored once and stored in the database. From this, it can be extracted and used as raw material for a wide variety of customized documents to meet the needs of vendors, partners, and customers. The benefit of this is that the publishing engine, in conjunction with the rest of the VIOMA components, extends the life of your corporate information by enabling it to be re-purposed.

## *Real-time data access*

VIOMA enables complete customization of documents in real time, so that documents can be dynamically assembled from a variety of sources, such as external databases, business logic modules, and document fragments from the database. Moreover, the publishing engine is able to recognize individual customers and target groups, and intelligently aggregate content to meet the needs of the audience. It matters little whether they are the latest sports results, stock prices or parts inventory, web documents can be dynamically constructed from a variety of sources to create documents in real time.

## *Profiling*

VIOMA provides a framework for customer profiling, which means that your company can track customer behavior and preferences in order to tailor the content into personal relationship marketing. Let us introduce Mr. Jameson: Mr. Jameson pays a call to Victoria Wines' website. His visit generates, for instance, the following information, which can be stored for later use in marketing activities.

- Which clarets caught his interest.
- The amount of time spent with Beaujolais and Rioja, respectively.
- The path he chose when navigating the website.

## *Security*

VIOMA supports high security functionality to ensure secure communications protocol used to encrypt/decrypt data during communication over the Internet. VIOMA supports encryption/decryption from 56 up to 2048 bits and can handle different types of algorithms.

VIOMA supports:

- TLS 1.0
- Security, SSL 3.0

SSL can be used at two levels:

- Plain https communication, where data is encrypted before being transferred over the Internet.
- Client Authorization, where additional security is added through the use of certificates, to verify that clients are authorized to communicate with the server. The highest level of security is offered through the use of a Certificate Authority (CA) server.

## *Middleware integration and data interchange*

With its wide range of connectors, VIOMA integrates easily with third-party middleware, thus laying the foundations for business-to-business communications. Using XML as the data interchange language, the connectors provide a powerful mechanism for storing and retrieving data between disparate business systems.

For example, VIOMA supports:

- A COM Connector for communication with Microsoft COM + distributed objects.
- An RMI/CORBA Connector, enabling you to tie VIOMA into enterprise solutions from vendors such as BEA Tuxedo, IBM, WebLogic, and Sun Communications.
- A TCP/IP Connector to store and retrieve content over a TCP/IP-session, which can be used to integrate third-party solutions written in C++, Java, C#, or Visual Basic.

## *Performance*

VIOMA provides extended functionality for fast and efficient delivery of information and for making the most of your company's IT resources. For example, VIOMA offers:

- Connection pooling for intelligent management of user connections to the system.
- Pre-compiled XML for fast loading and transformation.
- Load balancing for efficient distribution of resources and fast performance.
- Scalability for multiprocessor machines.

## *Vioma System Architecture*

A Vioma Server can be installed as shown below with two separate Java processes. In this setup, one primary process runs a Web server for the VCM interface, accessed via port 9600, and the Vioma publishing engine, while a secondary process runs a Web server to handle requests from the end user on the deployed site, via the default port, port 80. The second process uses the J2K web server interface to pass requests to and receive responses from the main process. Additionally, Java applets are downloaded and run to support some the functions of the VCM interface, such as the Authoring work area of the Content Author interface.

You can also install a Vioma server as a standalone server, without an external Web server attached. Vioma includes its own HTTP server that can be turned on for serving up content on either the standard port 80 or any other port desired (default is 8000)
The Vioma Server configuration shown above runs with two separate java processes. One java process runs the Vioma Publishing Engine and a Web server for the administrator's console, accessed via port 9600, which is used to create and manage content, XML templates, and business logic. Java applets are used to facilitate the user interface for the administrator's console
A second java process runs a Web server which is dedicated to serving web pages to the end user via the default port, port 80. This process uses the J2K web server interface to pass requests to and receive responses from the Vioma Publishing Engine.
The Vioma server can be configured to run with only one Java process. In this configuration there is no separate Web server exclusively for the end user. All requests, whether from the administrator's console or the end user, are handled via port 9600.

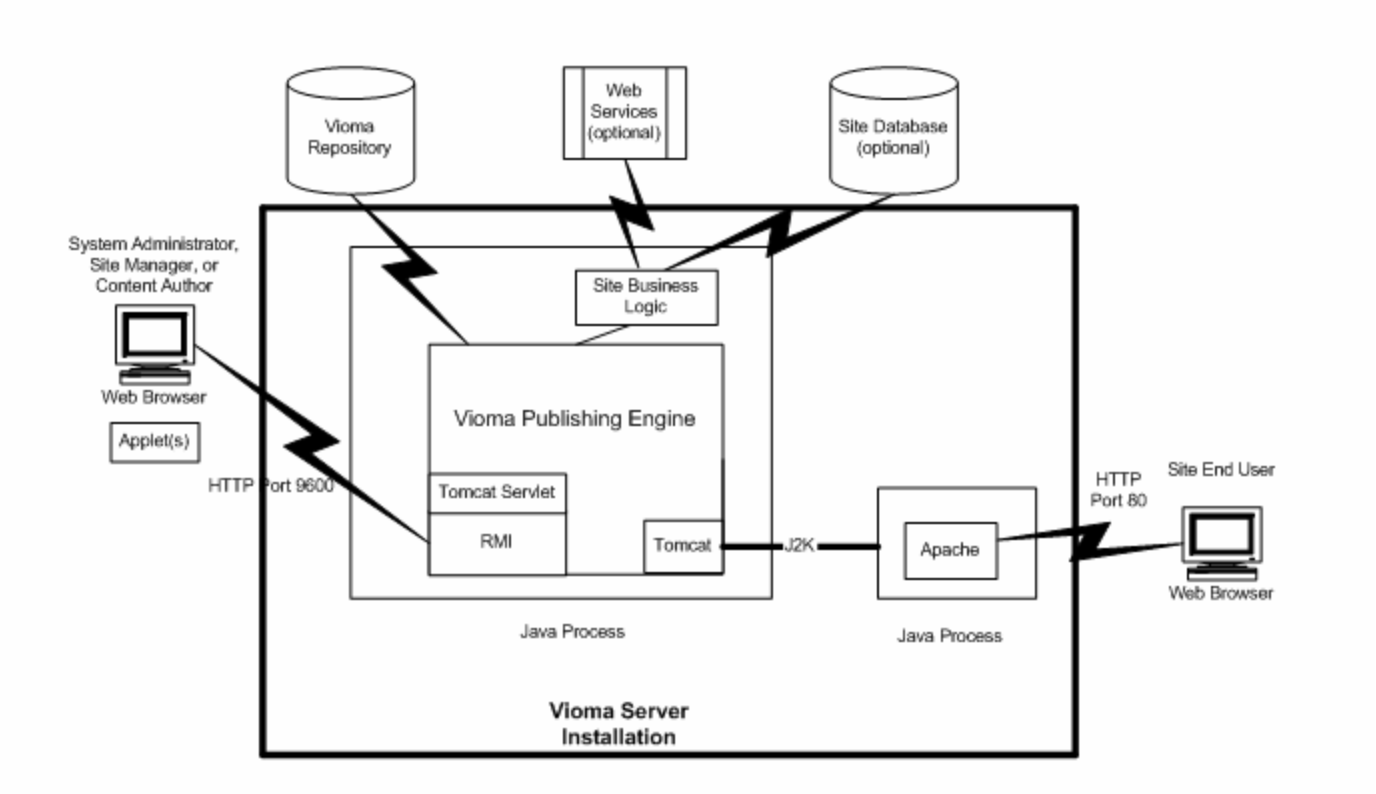
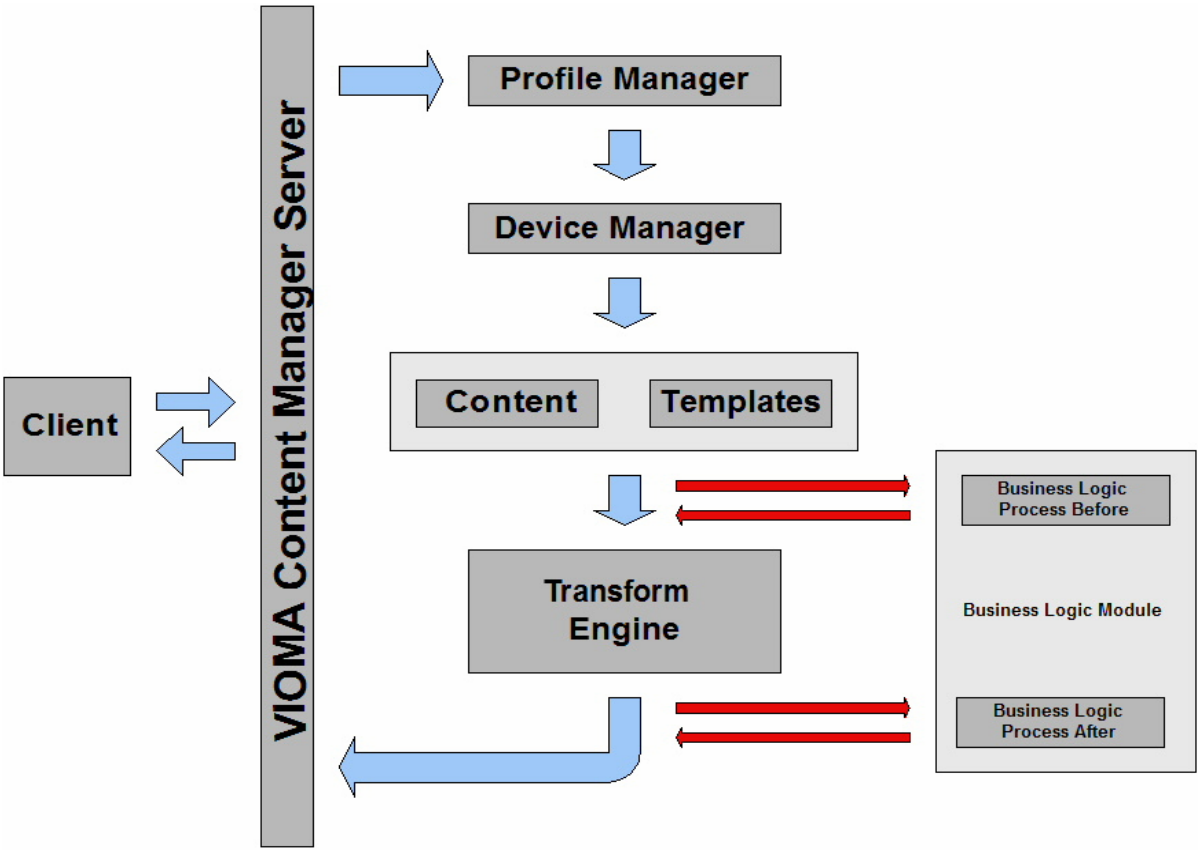



**Figure 1, Vioma System Architecture**

The diagram above shows the overall system architecture of Vioma Content Manager.

**This picture shows how information is dynamically processed by the server:**




When a page of content is created, it is stored as an XML document in the Vioma repository (content database).
Content is stored within a tree structure. Each node of the tree may be assigned a default template to be used when displaying the content within it. Each content item may override the default template.

When the page is requested for display to the end user, the proper template is retrieved and it is processed against the XML representing the content to produce an XHTML document as output, which is then processed by the transform engine and shown by the Web browser.

Templates can be created to provide the same information, shown differently, based on the receiving workstation's hardware capabilities, such as full-blown web browsers, PDA type devices, or cell phone displays.

This process may be customized for a particular site and node by implementing *processBefore()* and *processAfter()* interfaces provided by Vioma. As shown, if business logic is defined for a node, during the process illustrated above, Vioma will execute a processBefore() method, if one is implemented, before the template is applied to the content. This allows Java code to modify the XML document representing the content prior to its processing. If the business logic defines a processAfter() method, it is executed after the XHTML document is generated, but before it is served up to the end user. This allows Java code to modify

the output HTML prior to its display. See Section 6 of the *Vioma Education Program* document for more detail about business logic processing works.

## *Conclusion*

VIOMA Content Management Server ™ 4.0 gives you simplified site management, faster publishing times, flexible connectivity, reduced costs, and live documents constructed in real time. In addition, VIOMA protects the value of your information and positions your website to take advantage of emerging net technology. Perhaps most important, however, is the fact that VIOMA is easy to use, which means that it can be put to work immediately across the enterprise, thereby raising the productivity and improving the responsiveness of your website.

# Chapter 5: A primer in XML, XSLT, and CML

A paradigm shift is taking place on the Internet. At the head of this revolution is Extensible Markup Language (XML), the successor to the better-known HTML.

This chapter will introduce you to XML by producing a source document and at the same time demonstrate the advantages of XML when compared to HTML.

When the basics of information structuring have been described, it is time to demonstrate how an XML document can be transformed into another by using XSLT, an XML language for style sheets. To conclude, we will demonstrate a quick and easy shortcut to transform an XML document by using VIOMA's Content Markup Language CML.

This chapter is by nature technical; the illustrating code snippets have, however, been consigned to an appendix (see Appendix I - Code illustrations) in the hope that this will make the chapter, if not more enjoyable, then at least at little more digestible.

## *An introduction to XML*

XML has come out of obscurity. It has been called the lingua franca of the Internet and we are witnessing a paradigm shift in Internet communication. This is to a large extent true, but there is no lack of either exaggerations or uncertainties when it comes to discussing XML.

The major characteristic of XML is that it is not just the next generation of Markup Language or, to put it differently, an improved version of HTML. HTML defines a limited number of markers that describe an equally limited number of elements. Usually letting the markers describe how the document is to be displayed in a web browser does this. If there is no marker that describes the desired function, there is little you can do. Another limitation with HTML is that the markers describe presentation rather than content.

XML, on the other hand, allows the production of markers with the desired functions as work progresses. These markers also have certain limitations, but they are very flexible. Another important characteristic with XML is that this language describes the structure and meaning of a document. The fact that the presentation of the document is decided in a different part of the system means that the process has been streamlined to handle only information, not information and presentation.

XML markers describe the content of a document, not how it should look. For a more detailed description and explanation of this, turn to Appendix 1.

## *Transformation of XML with XSLT*

The simplest way to explain a style sheet is to say that it contains the instructions that govern the look of the information, a kind of graphic manual. XSLT is an XML-based language used for constructing style sheets. In this chapter, we will skip the technical details and focus on the results instead.

XSLT makes it possible to transform a document written in XML to a completely different XML document. It is therefore possible to create documents customized for different types of users or clients from the same source document. There is no longer any need to store one set of information for retailers with palmtops,

another for retailers with PC's, a third for retailers with..., etc. One set of information and the requisite number of style sheets is all that is needed.
Example 2 in the appendix demonstrates this by using XSLT to transform the source document "showtimes.xml".

In this code example, for-each statements are applied to nodes in the document "showtimes.xml" to extract information. When they match, the information is forwarded to the new document. What happens is that only the relevant information is forwarded and in this case the question of what is relevant is decided by the fact that the user is in Seattle and that this has been registered as a criterion for the search.

To apply this new style sheet, called "showtimes.xsl", we put a reference to it in the document "showtimes.xml" in almost the same way as you would add a cascading style sheet.

When we open the document "showtimes.xml", the new style sheet will transform it to show the showtimes in Seattle only.

This is only a very basic example of what can be done with XSLT, but hopefully it has given an idea of the inherent possibilities. For instance, XML documents can be transformed to show the same data in different ways, or information can be gathered from different sources to create new documents. Furthermore, different kinds of business logic can be applied to achieve even more precision in both the search and the presentation.

## *An introduction to CML*

Having shed some light on the mysteries on XML and XSLT, we have to ask: How do we store XML-based information in a database, and how do we get it out with a minimum of hassle? This is where Content Markup Language and VIOMA Content Management Server ™ come in. CML gives you the power and flexibility of XSLT, but hides much of its complexity. Using VIOMA, you can build two types of templates:

- XSLT-templates extended with CML tags.
- Simple CML templates.

We have seen a demonstration of how to create a style sheet with XSLT. Now we will see a demonstration of how to create style sheets, or templates, with CML. The fundamental difference between CML and XSLT is firstly that simple CML templates are HTML-based and secondly that the link between the source document and the CML template is placed by the Website administrator, rather than as a reference in the source document. This centralizes control over the look and feel of the website.

Let us refer to an illustration: With VIOMA system, it is possible to add a hyperlink to the source document "showtimes.xml" as shown in example 3.

Thus, it is very easy to transform information with CML without sacrificing the flexibility of XSLT, since CML markers can be added to style sheets written in XSLT. Perhaps most important is that CML has markers that simplify the work of extracting information from databases, no matter where they are.

## *Conclusion*

As we have demonstrated in this chapter, XML, XSLT, and CML offer enormous flexibility when it comes to

dynamically constructing documents. XML transforms data into pure information, XSLT formats and presents it, and CML simplifies the use of XSLT.

# Appendix 1 - Code illustrations

Example 1:

In HTML, a section heading for a document may look like this:
<H2>Showtimes</H2>

In the previous code fragment, the <H2> HTML tags contain formatting information. For example, the <H2> tags instructs the browser to display the showtimes information as shown in the following illustration:


But with XML, the same information can be crated with the following tags:
<SHOWTIMES>Showtimes</SHOWTIMES>

The tags are now self-describing. What is more, they contain no formatting information. In order to illustrate why this self-describing data is so powerful and why the separation of content from formatting offers such great advantages, we will flush this example out a bit more.

Many Internet portals provide movie times for different cinemas across the country. The problem is that cinemas in different cities show different films at different times, so that each city requires its own document. With XML, it is sufficient to create and maintain one single document. Then, based on the location of the user requesting movie information, data can be dynamically extracted from the source XML document to include only the target city. To begin with, let us have a look at how an XML source document might be constructed for two cities, Seattle and Bellevue. Each city, as is evident from the example, has a cinema or cinemas, and each cinema features several films with different show times:

```
<?xml version="1.0"?>
<SHOWTIMES>
<CITY>
        <CITY_NAME>SEATTLE</CITY_NAME>
        <THEATER>
                <THEATER_NAME>OPLEX</THEATER_NAME>
                <MOVIE>
                        <MOVIE_NAME>COOL HAND LUKE</MOVIE_NAME>
                        <MOVIE_TIMES>12:00;4:00;9:00</MOVIE_TIMES>
                </MOVIE>
                <MOVIE>
                        <MOVIE.NAME>BLADERUNNER</MOVIE_NAME>
                        <MOVIE_TIMES>12:00;4:00;9:00<AIOVIE_TIMES>
                </MOVIE>
        </THEATER>
        <THEATER>
                <THEATER_NAME>SHOWBOX</THEATER_NAME
                <MOVIE>
                        <MOVIE_NAME>FRIGHT</MOVIE_NAME>
                        <MOVIE_TIMES>12:00;4:00;9:00</MOVIE_TIMES>
                <MOVIE>
        <THEATER>
</CITY>
```

```
<CITY>
        <CITY_NAME>BELLEVUE</CITY_NAME>
        <THEATER>
                        <THEATER_NAME>CINERAMA</THEATER_NAME>
                        <MOVIE>
                                        <MOVIE_NAME>STARWARS</MOVIE_NAME>
                                        <MOVIE_TIMES>12:00;4:00;9:00</MOVIE.TIMES>
                        </MOVIE>
        </THEATER>
</CITY>
</SHOWTIMES>
```

Once the XML document is created and properly formed ( i.e. conforming to syntactical XML rules), the obvious question is: What can it do? At this point, if the document was to be opened in a web browser, it would look like this:

```
<?xmlversion="1.0"?>
<SHOWTIMES>
<CITY>
        <CITY_NAME>SEATTLE</CITY_NAME>
        <THEATER>
                        <THEATER_NAME>OPLEX</THEATER_NAME>
                        <MOVIE>
                                        <MOVIE_NAME>COOL HAND LUKE</MOVIE_NAME>
                                        <MOVIE_TIMES>12:00;4:00;9:00</MOVIE_TIMES>
                        </MOVIE>
                        <MOVIE>
                                        <MOVIE_NAME>BLADERUNNER</MOVIE_NAME>
                                        <MOVIE_TIMES>12:00;4:00;9:00</MOVIE_TIMES
                        </MOVIE>
        </THEATER>
        <THEATER>
                        <THEATER_NAME>SHOWBOX</THEATER_NAME>
                        <MOVIE>
                                        <MOVIE_NAME>FRIGHT</MOVIE_NAME>
                                        <MOVIE_TIMES>12:00;4:00;9:00</MOVIE_TIMES>
                        </MOVIE>
        </THEATER>
</CITY>
<CITY>
        <CITY_NAME>BELLEVUE</CITY_NAME>
        <THEATER>
                        <THEATER_NAME>CINERAMA</THEATER_NAME>
                        <MOVIE>
                                        <MOVIE_NAME>STARWARS</MOVIE_NAME>
                                        <MOVIE_TIMES>12:00;4:00;9:00</MOVIE_TIMES>
                        </MOVIE>
        </THEATER>
</CITY>
</SHOWTIMES>
```

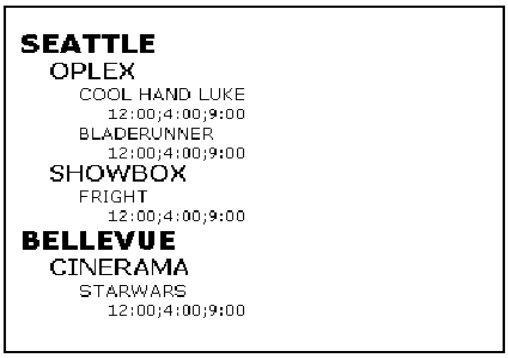
As you can see, it contains raw information, but no formatting information. One solution to this problem is to format the tags using cascading style sheets, the most commonly used method for formatting web pages today. We could, for instance, create a cascading style sheet in the showtimes.xml source document. It could look something like this:

```
CITY_NAME {display:block;font-size: 14pt;font-{amily:Verdana;font-weight:bold)
THEATER_NAME ldisplay:block:fo]it-size:12pt;font-family:Verdana;text-mdeiit:20px;font-weight:bold)
MOVIE_NAME (display:block;font-si2e:IOpt;font-family:Verdana;text-indent:40pxl
MOVIE_TIMES (display:block;font-size:8pt;font-family:Verda]1a:text-mdent:60px)
```

If this style sheet information is saved in a document called showtimes.css, we can add code to the XML document to attach the style sheet to it, as shown in bold in the following code:

```
<?xml versions" l,0"?>
<:?xml-stylesheet type="text/css" href="showtimes.css"!>
<SHOWTIMES>
        <CITY>
        <CITY NAME>SEA1TLE</CITY NAME>
```

Now, when the showtimes.xml document is opened in the browser, the following formatted document is displayed:




At this point it probably looks as if we have gone out of our way to create something in XML, which would have been much easier to create with HTML. But a look at XSLT will reveal how it can unleash the power of XML to create dynamic documents.

Example 2:

```
<?xml version="1.0'?>
<xsl:stylesheet version="1.0" xmliis:xsl=s"http://www,w3.org/1999/XSUTransform">
<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>
<xsl:for-each select="SHOWTIMES">
        Showtimes
</xsl:for-each>
<TITLE>
<HEAD>
<BODY>
<xsl:for-each select-"SHOWTIMES">
        <H1>SHOWTIMES<RI1>
        <xsl:for-each select="CITYICITY_NAME=Seattle'] ">
        <H2><xsl:value-ofselect="CITY_NAME"/><RIZ>
        <xsl:for-each select="THEATER">
        <P><wl:value-of-select="MEATERNAME"/>c
        <table border I" width="300">
                <thead>
                <tr>
                        <th align="left">Movie</thxth align="left">Movie Times</th>
                </tr>
                </thead>
        <tbody>
        <xsl:for-each select="MOVIE">
        <tr>
                <tdxxsl:value-ofselect="MOVIE_NAME"/></tdxtdxxsl:value-of select="MOVIE_TIMES"/x/td>
        </tr>
</xsl:for-each>
</tbody>
</table>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</BODY>
<HTML>
</xsl:template>
</xsl:stvlesheet>
```

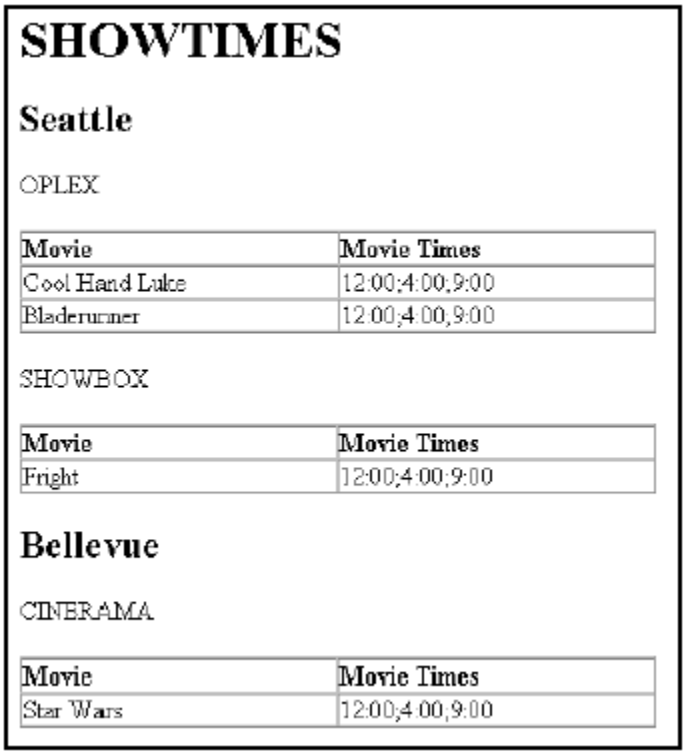
Example 3:

```
<AHREF="/servletA?bcs?action==get&robject=movies:showtimes"
```

This code fragment makes the "movies document", or the code fragment, the active object. In the template that formats the page, a reference to the body part of the document is specified, as is shown in the following code:

```
<HTML>
<BODY>
<cml:value-ofselect="contentbody"/>
<BODY>
</HTMI >
```

With the previous template, the body part of the "showtimes" source document will be incorporated into the HTML document, to produce the page shown in the following illustration:



# SHOWTIMES

## Seattle

OPLEX

| Movie | Movie Times |
|---|---|
| Cool Hand Luke | 12:00;4:00;9:00 |
| Bladerunner | 12:00;4:00;9:00 |

SHOWBOX

| Movie | Movie Times |
|---|---|
| Fright | 12:00;4:00;9:00 |

## Bellevue

CINERAMA

| Movie | Movie Times |
|---|---|
| Star Wars | 12:00;4:00;9:00 |

# Appendix 2: Using Barracuda's FormMap to Access and Modify Values

One of the primary advantages that Vioma Content Management Server 4.0 ™ has over simpler systems is its support for the execution of custom business logic to process Web pages. Development and processing of forms is generally one of the more difficult parts of website development. Vioma has libraries to allow a website to use the FormMap functionality of the Barracuda project (http://www.barracudamvc.org/Barracuda/javadocs/org/enhydra/barracuda/core/forms\/FormMap.html). The FormMap facility supports a database-driven architecture for a website by providing an easy way to map form values to a data base and provide validation. The information below provides some technical, code-specific detail about Vioma's support for and interface to the Barracuda Form Map libraries.

For a particular user, we might want to populate the form with the values from the data base (values previously entered) and fill the data base with values the user just entered. The same value that identifies the parameter (the name) can be used, of course, to identify the HTML DOM element which is modified. In other words, if the form looks like so:

<input name="address" type="text" />

we can modify the form by grabbing the element using XPath of "//input[@name='address']" and grab the value posted by getParameter("address"). It might be good to figure out exactly how to modify the element by determining the type and element name but it's probably quicker just to specify the Html type. Since barracuda already used the word HtmlType to specify the data type (e. g. String, int, etc.) we chose the word HtmlFormType.

So, at this point, with a little XPath work, we had a pretty tricky way of modifying the XHTML and processing the form with one definition. All that was needed was to hook up the value with the database. Since we were using Torque, most of the values had nice getters and setters. For example, once we got the user object, we could get the address from the call "user.getAddress()" or set it with "user.setAddress(String)".

Rather than lock into using XPath however, we defined an interface called IDataObjectModifier. This interface is used to modify the storage based on the values POSTed and access a value from the storage. The two methods are called setDataObject and getDataObject. The most common implementation of this interface is the XPathDataObjectModifier. We ended up extending barracuda's FormElement to include this functionality.

So, with a few utility methods, we define an element like so:

defineElement("address1", HtmlFormType.TEXT, newNotNullValidator("Please enter an address"), FormType.STRING,"addressLine1");

When the calling program first loads up the page, it creates an XPathDataObjectModifier object, handing it the XPath ("addressLine1"). It calls getDataObject to grab the value. Then it calls HtmlFormType.TEXT.alterDom() passing in the base object (the specific Torque User object), the htmlname ("address1") and the value. Now the user sees the same values in the form that are stored in the data base.

When the user posts a new value, the Barracuda validator (NotNullValidator) is called. If everything is OK, then we grab and translate the value that was posted (in this case, it is a String so there isn't any translation). We take that value, create another XPathDataObjectModifier object (using "address1") and call setDataObject().

## Explanation of terms used

**Clients**. Various kinds of hardware used for gathering information from the Internet. Among the most common types of clients might be mentioned P's and WAP telephones.

**Content**. A core concept. Content is the sum total of company information, internal as well as external, for instance links to external databases or information in business systems. Thus, the term content is not limited to the information that exists, or is about to exist, on the Internet. From this it follows that a content manager must be able to handle all content, regardless of media or channel.

**Content Author**. Content authors enter and maintain the information published on the site. Certain users may also be given control over what is published, by means of approving information using the built-in workflow control function. Authoring and workflow control is carried out in the Content Author module.

**Document**. Not exactly a sheet of paper. Document in this context means the final presentation of any given information. This does not mean that a document always looks presentable, since formatting instructions may be missing. A document is information that has gone through all the transformations that current rules prescribe.

**DTD**. (Document Type Definition) Determines the structure of an XML document (or set of documents). The DTD is defined as a part of the XML specification.

**Dynamically constructed**. Means that the information is adapted to the receiver and the amount of information available at the time of request. This means that the same question does not always get the same answer. The answer depends on who asks and when the question is put.

**Schemas**. A schema is an XML-based alternative to DTD. An XML schema describes the structure of an XML document. It defines the structure and the type of contents that each data element within the structure can contain.

**Site Master**. The Site Master sets up and maintains the information structure. The Site Master creates forms, DTDs, templates, user accounts, workflow, virtual URLs, and stages. This work is carried out in the Site Master module.

**System Administrator**. The system administrator sets up hardware, software, and communications. The system administrator is also responsible for the integration of VIOMA with surrounding applications and devices. System administration work is carried out in the System Administrator module.

**Website administrator**. A person with overall responsibility for the website, both when it comes to development and function.

**Web master**. A person working with tasks more or less directly related to the function of the website.

**Web designer**. As the name implies, a web designer designs the website, both the visible features and the features that work behind the scenes.