6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

MythoLogic: problems and their solutions in the evolution of a project

István Székely[a], Róbert Kincses[b]

[a]Department of Information Technology,
University of Debrecen
e-mail: iszekely@inf.unideb.hu

[b]Department of Information Technology,
University of Debrecen
e-mail: kincsesr@inf.unideb.hu

Abstract

The MythoLogic project started as an R&D project in May 2000. It was initiated by the University of Debrecen and the T-soft Engineering Ltd. with some students involved. In the beginning the objective of the project was to create a portal which provides an opportunity to individuals, companies, organisations, etc. to appear on the Internet. The portal was built for those who do not have the possibility to maintain their own information infrastructure. The main advantage is that the portal has an editor application which enables the entities to edit and publish events, news and other specific information using only a web browser.

The first implemented version, called Digitalcity, was a commercial product and has been used in Hungary (Debrecen). It was only a single portal and it supported only one language. The portal became more popular so the requirements had changed quite a lot. The latest version of the MythoLogic handles multiple languages, supports several portals in a specific domain, and is available under different domain names. It has been sold not only in Hungary but in Sweden, as well.

During the four years of development, our team obviously had to face several problems. In this paper we would like to introduce the evolution of the project, the problems which have arisen and the ways we solved them.

Categories and Subject Descriptors: H.3.5 [Information Storage And Retrieval]: Online Information Services; H.4.3 [Information Systems Applications]: Communications Applications - *Information browsers*;

Key Words and Phrases: Web application, Web-based services, Web portal development

1. Introduction

The MythoLogic project started as an R&D project in May 2000. It was initiated by the University of Debrecen and the T-soft Engineering Ltd. with some students involved.

In the beginning the objective of the project was to create a portal which provides an opportunity to individuals, companies, organisations, etc. to appear on the Internet. These are collectively called entities. These entities can edit and publish information to the Web using solely the tools provided by MythoLogic.

1.1. Parts of the application

The MythoLogic application is composed of three discrete modules. In the following the parts of the application will be described briefly.

1.1.1. The viewer

The first part of a MythoLogic portal is similar to a conventional portal. This part displays specific information about the entities. Information is arranged into lists in which one can turn over pages. Clicking the list items causes detailed information to be displayed.

One can find the main menu on the top of the page. This menu determines the type of items appearing in the list, for instance, entities, events, news, and so on. There is a menu on the left side and each menu item may have submenus. One can think of this menu structure as a tree. It filters the list displayed in the middle of the page.

There is a drop-down list that contains category names (catalogue). This list means another point of view filtering the elements of the list. The visitor can use this list to look up, for instance, car mechanics. The entities can choose few categories, in which they will be included.

There is a search engine which helps to find the necessary information without navigating through the menus or selecting any category.

1.1.2. The digital map

Every portal has a digital map that can show all the entities. The pages containing the map also have a menu. Entities can be selected using this menu.

The map is capable of showing geographical location of the entities and further information can be extracted from the map. It can be done by clicking on an entity or selecting a group of them. The information set obtained using the map is the same as using the viewer module.

1.1.3. The editor

The portal was built for those who do not have the possibility to maintain their own information infrastructure. The main advantage is that the portal has

an editor which enables the entities to edit and publish events, news and other specific information using only a web browser.

The viewer and the map modules are similar because they both take care of displaying information and belong together. The editor differs from them because the main purpose is to create new and modify existing information, so it can be called rather an application not a module.

The editor application also has a menu bar but it differs from the menu of the viewer module. This menu looks like those which are used in common desktop applications. The menu items can have submenus. The menu items can be selected by mouse which results in showing up dialogs. The more frequently used functions are available from a tool bar.

2. The way of the evolution

The MythoLogic application went through several stages before it has reached its recent form. The look of the application and the implementation have been changing for four years. Its functions have improved a lot during the advancement. This section will outline this process.

2.1. One portal, one language

The first implemented version was the Digitalcity Debrecen portal. It was introduced as a commercial product at the end of 2000. It was intended to be the official portal of Debrecen.

The application dealt with only a single portal and it supported only one language and thus perfectly fulfilled the requirements. This task could be accomplished using Net.Data which is a macro processor developed by IBM, but the map module was written in Java language from the beginning.

2.2. Working with multiple portals

The portal became more popular so the requirements had changed quite a lot. The application had to be developed in a way that it could handle multiple portals but the integrity of the information had to be kept intact.

We had to ensure the navigation between the individual portals. Besides that it was necessary to control that whether an entity can appear on a portal or not. Furthermore, the rights of editors had to be constrained in order to guarantee that a certain editor can modify only the information that possessed by him/her.

The tools to build the applications remain the same.

2.3. Handling different languages

The previous version, mentioned in the subsection above, unfortunately still suffered from being able to work with only one language. One portal was made for

Hajdúszoboszló and after a short period of time a demand emerged which requested the portal also in English.

We had to find a solution to solve the problem of handling different languages (not only English). Handling multiple languages means handling several character coding systems. We managed to adapt the application for national language support, but Net.Data lacks appropriate functionality. Characters from different code pages could not be displayed on a single page (for example the *å* from the Latin-1 code page and the *ő* from the Latin-2 code page). Therefore, we switched to Java and the viewer module was rewritten using it.

2.4. The recent version

The latest version of the MythoLogic application supports several domains. Each domain consists of a small number of portals. Many of the portals are available in various languages.

It has been sold not only in Hungary but in Sweden, as well. The working version can be seen at [1]. MythoLogic is developed and improved continuously. Now the editor application is being reimplemented in Java because of its features.

3. Problems and their solutions

During the four years of development, our team had to face several problems. We will describe the problems which have arisen and the solutions we found. In the following, the tools used in development and their drawbacks will be discussed. We will explain briefly why we have come to the decision to ignore some of these tools and to use others.

3.1. Net.Data

Net.Data is a macro processor that executes as middleware on a Web server machine [2]. It is capable of creating HTML pages dynamically. It used to be a very important tool for generating dynamic content (see 2.1).

3.1.1. Essentials

The basic unit is the macro. In a macro variables, statements and function calls can be used—like in another programming languages. A macro is a simple text file and consists of a declaration and a presentation part.

The declaration part can contain variable declarations, function and macro function definitions. Functions can be written using external language, for instance SQL. These elements can be included from another file.

The presentation part is made up of HTML or XML blocks. Each block must have a name which is unique inside the macro. Each block represent a web page which will be sent to the clients. In fact, the name of the blocks are the entry

points of the macro, so every macro has to contain at least one HTML or XML block. In order to generate a web page, one has to supply the name of a macro and the name of an entry point.

These blocks contain the description of the pages. In addition to that, they can contain variable references, function and macro function calls, include statements and control flow statements. Anything which is not recognised by the macro processor sent to the client without modification, so we can put HTML and JavaScript elements into our blocks.

3.1.2. Disadvantages

Net.Data uses an interpreter to process the macros and generate output. The drawbacks of interpreters are widely known, such as it interprets only one line of the code at a time, and there may remain syntactic errors in programs because of inadequate testing. When multiple language support had been introduced (as we mentioned in 2.3), it caused the creation of a large number of language variables. These variables were stored in text files as Net.Data variables. One of these files had to be included in every macro and as the number of variables increased, the performance dropped significantly.

In addition to the performance issues there is an another problem regarding to the language variables. Net.Data could only display pages made up of characters from a single character set.

Another problem was that Net.Data based on the HTTP [3]. It is a generic, stateless protocol, and Net.Data contains nothing to support handling user information. It does not provide tools for working with sessions, at all.

Finally, IBM discontinued to support it.

3.2. WebSphere

WebSphere is a servlet and JSP container [4]. A *servlet* is a Java programming language class which extend the capabilities of servers [5]. It is a general-purpose server-side technology, however, Java Servlet technology defines HTTP-specific servlet classes. The *JSP technology* is an extension to the servlet technology [6]. JSP enables all the advantages of servlets. Moreover, it eases creating static content in the pages at the same time.

Java and therefore servlets are based on the Unicode character encoding standard [7], which encompasses all the conventional character sets like ISO Latin-1 and ISO Latin-2. It solved the character set problem mentioned in 2.3.

The servlet technology provides so-called scope objects. One can think of them as a special storage place. There is one scope object at the level of the application and every session has one. Using them it is possible to emulate a stateful communication over HTTP. Any kind of information can be stored in these objects.

In the editor application (1.1.3) is was necessary to keep the creation and destruction of sessions under control. We needed fine-grained control of sessions,

but the WebSphere provided only Servlet API version 2.2, which did not make it possible. Certainly, a new version of WebSphere is available, but it is too expensive.

Nevertheless, the IBM WebSphere offers indisputable advantages compared to the Net.Data. Besides that, it has some drawbacks which compelled us to change to Tomcat.

3.3. Tomcat

Tomcat is an open-source servlet and JSP container developed by a community of volunteer developers and managed by the Apache Software Foundation [8]. It is fully compliant with the Servlet and JSP Specification which is proven by the fact, that the servlet and JSP container of Sun's official J2EE Reference Implementation based on the Tomcat.

Contrary to WebSphere, the Tomcat versions are released frequently, so it is always up-to-date which means it implements the newest technologies and standards. Furthermore, bugs are corrected in a short period of time after they have been revealed.

It is implemented in pure Java, which results in two important things. First, the underlying JDK along with the Java Virtual Machine can be changed without serious complication. The recent Java API can be used. Second, it is truly platform independent, hence, it can be used on AS/400, as well.

3.3.1. The utilised benefits of the Tomcat

It has turned out, that the overall performance is perceptibly better than using Net.Data. This performance gain is achieved by the following features made available by Java and the technologies implemented by the Tomcat.

The database contains a large amount of data which changes rarely. It is very time-consuming to query this kind of data straight from the database every time when it is necessary. This data can be stored in and retrieved from a cache. The cache is implemented as a bean with application scope. In the MythoLogic web application, for example, there are catalogues whose items can be cached.

Another possibility to improve performance is to reduce the time of opening a connection to the database. We have implemented a connection pool, that stores a collection of opened connections. When a JSP has to execute a query, it gets a connection from the pool. If no connections are available, the pool will open a new one and returns it to the JSP. After executing a query, the connection must not be closed but has to be returned to the pool.

In MythoLogic, there are a hundreds of queries. In order to ease the management of queries, they are kept separated from the source of pages in XML files. There is a query manager, which parses the XML files and keeps all the queries in memory. Every query has a name which identifies it and can be retrieved from the manager using its name. An XML file contains not only the name of a query but the documentation that belongs to it, as well.

The Lightweight Directory Access Protocol (LDAP) is a general-purpose protocol to access data stored in directories which are structured hierarchically, like Novell Directory Service or Microsoft Active Directory. In a directory, names are associated with objects. The Java platform contains the JNDI API, which enables us to access several naming and directory services, including LDAP. Information about the users of MythoLogic editor application, like user name, encrypted password and authorisation information are stored in the IBM SecureWay Directory. During the login we look up this information using LDAP.

4. Summary

The MythoLogic project started in 2000. Requirements have been changing since then. It was initiated as an R&D project but is evolved into a commercial product.

The first version of MythoLogic was created using Net.Data. It was appropriate for implementing the product rapidly but because of its disadvantages we have to use different tools. Our intention is to ignore the Net.Data completely.

Now we are using Java, Tomcat and related technologies like JDBC, JSP, servlets and JNDI. The editor application is being reimplemented in Java. After this process the Net.Data can be eliminated.

The MythoLogic web application is evolving continuously and we are eager to develop further. We might state that the MythoLogic is successful, since it has been sold and is in use at several companies and organisations.

References

[1] *The MythoLogic portal powered by MythoLogic application*, `http://www.mythologic.hu`

[2] IBM Corporation, *IBM Net.Data Administration and Programming Guide*

[3] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*, IETF Network Working Group, 1999.

[4] IBM Corporation, *WebSphere Application Server*, `http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=products/appserv`

[5] Sun Microsystems, Inc.: *Java Servlet Technology*, `http://java.sun.com/products/servlet/index.jsp`

[6] Sun Microsystems, Inc.: *JavaServer Pages Technology*, `http://java.sun.com/products/jsp/index.jsp`

[7] The Unicode Consortium, *Unicode Standard*, `http://www.unicode.org/versions/Unicode4.0.0/`

[8] Apache Software Foundation, *The Tomcat 4 Servlet/JSP Container*, `http://jakarta.apache.org/tomcat/index.html`

Postal addresses

István Székely
Department of Information Technology
University of Debrecen
P.O. Box 12, 4010 Debrecen
Hungary

Róbert Kincses
Department of Information Technology
University of Debrecen
P.O. Box 12, 4010 Debrecen
Hungary