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

The mobiDIÁK Self-Organizing Mobile Portal*

Péter Antal, Norbert Bátfai, Péter Jeszenszky

Department of Applied Mathematics and Probability,
Institute of Informatics, University of Debrecen
e-mail: antalp@inf.unideb.hu

e-mail: nbatfai@inf.unideb.hu

e-mail: jeszy@inf.unideb.hu

Abstract

MobiDIÁK is a portal engine that can be used as a basis to build self-organizing mobile portals. Mobile means that portals that are built on the top of this engine support mobile client devices such as WAP-enabled mobile phones or PDAs as well as traditional web browser clients. Self-organization means that it provides the community of users with the ability to organize themselves and to run their own portal.

In this paper we give an overview of the architecture of the portal engine and the services that it provides. We also discuss some implementational details, such as the techniques and technologies used in the development.

Categories and Subject Descriptors: H.3.5 [Information Storage and Retrieval]: Online Information Services - *Web-based services*

Key Words and Phrases: Web development, portal engine, mobile technology, community building, server-side Java technology

1. Introduction

The goal of the project is to develop a portal engine that enables us to create self-organizing mobile portals. Self-organization means that portals built on the basis of this engine provide functions to their user community

- to develop the content of their portal according to their needs,

*The development of the mobiDIÁK portal engine and reference clone is supported by the Hungarian Ministry of Education and by the Hungarian Ministry of Informatics and Communications under grants IKTA5-141/2002 and ITEM50/2003.

- to organize their activity,
- to maintain the operation of their portal.

Mobile means that most of the services of these portals are also accessible to mobile devices, such as WAP-enabled mobile phones and PDAs.

To be more specific we focus on a special domain: higher education. Based on the portal engine developed we have also built an operating portal that we call *reference clone*. The reference clone demonstrates the possibilities (and also the limitations) of the portal engine behind it.

Since it is an open source project the developed portal engine is available under the GNU GPL license. The reference clone is also open to the public, but using much of the services requires free registration.

2. The elements of the portal model

2.1. Users

Designing a special purpose portal engine we tried to mimic the hierarchical structure of the target domain i.e. higher educational institutes. Identifying the most typical roles we have developed a structure called the *pyramid of skills* that constitutes the basis for the portals operation.

Each level of the pyramid corresponds to a set of users with a specific role. The levels of the pyramid are the following (from bottom to top): unregistered users, registered users, students, expert students, experts, and leader experts.

Each level has its own well defined privileges. Users contribute to the building of the community with their activity through the portal services. The upper levels of the pyramid are supplied with the most wide range of privileges that is they play a central role in self-organization.

The expansion of the user community also relies upon the pyramid. Registration as a registered user or student is available without any restrictions. Registration as an expert student or expert requires the permission of some members of the community that have the privileges to approve the registration. (During the registration procedure it is necessary to name the members that should confirm the registration.) The leader experts at the top of the pyramid are registered by a special registration method using a VIP code.

2.2. Documents

The other elements of the model are documents. The portal engine itself ditinguishes a wide range of documents (e.g. textbooks, lecture notes, bibliography collections, link collections and downloadable software). The portals offer their documents together with metadata as content available to the public.

2.3. The interaction of users and documents

The interactions occur when users access the documents offered by the portal engine. Each user and each document is assigned a numeric vector that measures its relevance. The components of the vectors correspond to predefined categories. Each component is a non-negative number that tries to express the relevance of the user or document in the corresponding category. (The larger the value, the larger the relevance.) As users and documents interact their vectors change dynamically according to the built-in rules of the engine.

At the present time a simple mathematical model constitutes the basis of the dynamics. This model relies upon the following assumptions.

- Documents become more relevant as the more number of relevant users access them.
- Users become more relevant accessing relevant documents.

We are experimenting with the described dynamics. The results will be published later.

3. The reference clone

The reference clone is a portal on Informatics[1]. Its primary goal is to serve as a forum where people learning and teaching Informatics can meet, and also displays content that our home institute offers to the public.

Most of the currently available content has been developed as a part of the project. The content developers – they are typically lecturers of our home institute – have produced 37 items that are freely available at the portal. Among these there are textbooks, exercise books and other course materials. They cover different topics including Calculus, Discrete Mathematics, Neural Networks and many others. Each course material contains a large amount of information, their average length is 150 pages. This internal series of documents on our portal is called *mobiDIÁK Library*.

Most of documents are prepared in LaTeX, thus the portal typically offers them in different formats like Adobe Portable Document Format and PostScript. Special documents such as editable bibliography and link collections are rendered as HTML pages.

In addition to the reference clone there is also a test clone that demonstrates the most current status of the project[2]. Since the test portal is used to test new functionality and many of its services are still under development it is not intended for public use. New services are added to the reference clone after extensive testing.

[1] See: `http://mobidiak.inf.unideb.hu/`

[2] See: `http://mobidiak.inf.unideb.hu/mobidev/`

4. Portal Services

Portal services can be accessed by different client devices.

- Almost every service is accessible by a HTML based web browser.
- Many of the services are accessible by a WAP-enabled mobile phone. Due to the limited capabilities of this kind of devices the portal engine provides them with a small subset of the content compared to traditional web browser clients.
- Almost every service is accessible by a PDA (such as a Palm).

Using some services might require privileges.

4.1. Core Services

The portal engine supports the following services as core services. These are called core services since they are deeply integrated into the engine and any portal clone built on the top of the engine will provide them.

4.1.1. User management

- Registration. Registration might require the permission of some members of the community. That is the community itself controls its own expansion.
- Login. Using much of the services requires users to login.
- Personal homepage. Each user owns an automatically generated public homepage that contains personal information provided during the registration procedure. Documents belonging to the user are also accessible from here.
- Directory. The directory contains all of the users in alphabetical order. The personal homepages can be accessed from the directory. There is a seperate directory for students and another one for teachers.
- Search. Search the directory for students and teachers.

4.1.2. Document management

- Upload document. Each document is a set of downloadable files together with descriptive metadata. This means that electronic material can be provided in differents formats and in multiple parts, and these are all available under the same title.
- View or download document. Documents are available for viewing and downloading to anyone without restrictions.

- Modify document. The user who owns the document can modify its metadata and the downloadable files that it is made up of, or delete the whole document.

- Search. Documents belonging to a specific user can be accessed directly from the personal homepages. Documents satisfying a search criteria can be retrieved using the metadata based search facility.

4.1.3. Messaging

Users can communicate using the internal messaging system. The engine provides the necessary functions to read, write and delete messages.

4.1.4. Exam results

Exam results of students can be published at the personal homepages. The engine provides the necessary functions to create, edit, delete and view exam results.

4.1.5. Forum

The usual forum facility. In our opinion forums might be used heavily in student-teacher communication (e-consultation).

4.1.6. News

The portal provides the community with the latest news that they might found interesting, including the news about the development of the portal engine, community life and events.

4.2. Additional Services

These services are not integrated into the portal engine. Instead they are implemented as a part of the reference clone.

4.2.1. KódBazár

The online electronic journal of the reference clone. Its mission is to popularize Java technology and Java technology based solutions aiming to cover a wide audience. The journal is divided into sections that offer articles for IT managers, programmers as well as for outsiders.

4.2.2. MIDletTár

This facility enables the users to publish their MIDlets (Java applications for Java-enabled mobile devices) at the reference clone thus make them available to the public.

5. The Phases of Development

The conceptional design of the project is presented in the following document that is the result of the first phase (phase 1) of the development:

`http://mobidiak.inf.unideb.hu/dev/koncepcio.html`.

Based on this document we have developed the object model of the project in parallel with specifying the required functionality of the portal engine and also the user interfaces (phase 2). The implementation has been carried out in the next phase (phase 3).

6. The Architecture of the Portal Engine

Since the main task of the project is to develop a portal engine that supports self-organization and a reference portal on the basis of the engine, the main focus is on the server-side. Our goal was to develop a portal engine that

- has a simple, transparent and modular architecture,
- supports different client devices (web browsers, mobile phones and PDAs),
- supports the addition of new functionality on the fly (since the duration of the project is three years, we extend the engine and the operating reference clone with newly implemented functionality as they have been done),
- further improvements should be possible.

We have separated the Controller, View and Model components of the server-side according to the Model-View-Controller design pattern. The Controller component has been implemented by an abstract class named MobiServer. The View components are special servlets (see later). The Model components have been implemented in a separate package (separate from the Controller and View components).

The MobiServer abstract servlet class provides the following functions.

- Client authentication. It is the basis of user management. When a client authenticates itself to the server the later creates an object that holds connection information and exists until the connection is closed.
- Permission management. Since the portal engine provides a wide range of functions and each level in the pyramid of skill has its own well defined privileges, using the portal services requires a permission check.
- Data validation. It is the syntactical and semantical validation of user input. On erroneous input the user is provided with the necessary instructions to recover from the error. Since each servlet implementing a portal service imposes its own constraints on user input the method that corresponds this function is abstract.

- Measuring relevance. A separate module handles the interaction of users and documents that processes the requests of the Controller asynchronously.
- Response generation. When a client request generates an error or warning, the client is provided with the necessary instructions to recover from it. The View components are responsible for providing this funcionality that is why the corresponding method is abstract.

The View components are special servlets that are subclassed from the MobiServer class. They implement portal services by imposing constraints on the input accepted and by providing the clients with the appropriate response based on the input and the Model. This means that they implement the abstract methods of their parent class. The implementation presented here does not conform to the well known Model 2 architecture since special servlets are used instead of JSP pages. The View components provide the following functions.

- Data validation. (See above.)
- Response generation. To generate response a simple cache technique is used. There are response templates that are read from the filesystem on startup. To form complete responses that might be sent to a client information items are inserted into the templates dynamically.

The portal engine supports different client devices and separate servlets being responsible for serving each type of device. Servlet classes handling a specific device are bundled into a package (web, wap and pda packages).

In summary, the Controller and View components are separated both logically and on the source code level. At the same time they are related through inheritance thus there is no need for forwarding the requests that are addressed to the server. Users on the client side "implement" this functionality by their mouse clicks.

7. Technologies Used

The user interfaces conform to the following specifications: HTML 4.01, CSS2, WML 1.2.

We have implemented server-side functionality using Java servlets that conform to the Java Servlet Specification 2.2.

We have used a number of 3rd party tools and class libraries in the development such as the Apache Ant build tool, the Xalan-Java XSLT processor, the Log4j API and a servlet package from O'Reilly.

The persistence layer has been implemented using the Castor data binding framework and is clearly separated from server logic.

Our underlying database product is PostgreSQL 7.4.1 but we do not take advantage of its implementation specific features thus the portal engine should also work on the top of any other database product supported by Castor (e.g. MySQL, Oracle).

We use the combination of the Apache web server (version 1.3.26) and the Tomcat servlet container (version 3.3) as an application server.

8. Testing

We have tested the functions of the portal engine using the following browser software: MS Internet Explorer 6.0, Netscape 7.1, Mozilla 1.6, Opera 7.23, Wapaka 3.18, Nokia Mobile Browser Simulator 4.0.

We have also used the following mobile devices for testing: Motorola T720i, Dell Axim X5, Palm Tungsten C.

The portal services are accessible by all of the mentioned software and hardware clients and user interfaces behave similar on devices of the same type.

9. The Current Status of the Project

The duration of the mobiDIÁK project is three years. Now we are at the end of the first year of the development and have already released a portal engine that offers a wide range of services. The functionality provided by the engine and thus the reference clone is growing. An early release of the portal engine serves as a basis for the homepage of our home institute.

10. Developers

The conceptional designs of the portal engine and the reference clone are the result of the work of a team that is comprised of the following members: Péter Antal, Norbert Bátfai, István Fazekas, Péter Jeszenszky. The implementation has been carried out by Péter Antal, Norbert Bátfai and Péter Jeszenszky.

Postal addresses

Péter Antal
Institute of Informatics
University of Debrecen
4032 Debrecen, Egyetem tér 1
Hungary

Norbert Bátfai
Institute of Informatics
University of Debrecen
4032 Debrecen, Egyetem tér 1
Hungary

Péter Jeszenszky
Institute of Informatics
University of Debrecen
4032 Debrecen, Egyetem tér 1
Hungary