mCMS: A Content Management System Adapter Architecture for Mobile Devices

Mark Joselli
MediaLab, IC-UFF
http://www.ic.uff.br/~medialab
Niteroi, RJ
mjoselli@ic.uff.br

Eduardo Soluri
Nullpointer Tecnologia
http://www.nullpointer.com.br
Rio de Janeiro- RJ
esoluri@nullpointer.com.br

Jose Ricardo Silva Junior
MediaLab, IC-UFF
http://www.ic.uff.br/~medialab
Niteroi, RJ
jricardo@ic.uff.br

Marcelo Zamith
MediaLab, IC-UFF
http://www.ic.uff.br/~medialab
Niteroi, RJ
mzamith@ic.uff.br

Esteban Clua
MediaLab, IC-UFF
http://www.ic.uff.br/~medialab
Niteroi, RJ
esteban@ic.uff.br

ABSTRACT

Most of the content on the World-Wide Web is designed for desktop computers. Nowadays, with the evolution of mobile phones, smartphones, tablets and Digital TV, this content must be adapted to these different devices, where each have specific characteristics and constraints, like screen dimensions. This adaptation task can consume time and computationl resources, since most solutions resolve it by creating different content versions for different devices. A CMS (Content Management System) is a software that keeps track of every piece of content that is used by websites and portals. A CMS can be adapted for others consumers, like mobile devices. This adaptation can be done by developing plugins for each different device, or creating different resources for each different device, because most of the adaptation needs to be done for each device. In this work, a novel CMS adapter architecture is presented, which is made to adapt the content on CMSs for different devices, through the use of templates that describes how the content must be transformed. Also, since most applications need some form of offline content and normally the cost of network is high, the architecture also provides cache management layer on the client side, in order to provide offline cached content, a data compression mechanism for keeping the data exchange in the network low, and a content version control. This way, the architecture avoid higher data transfers throught the network which in some cases can be slow and expensive over mobile networks.

Categories and Subject Descriptors

D.2.11 [[**SOFTWARE ENGINEERING**]]: Software Architecture

General Terms

Architecture, CMS

Keywords

CMS, Content Management Systems, Content Adaptation, Multimedia, Mobile Devices, Software Architectures

1. INTRODUCTION

Mobile devices, like smartphone, tablets, and Digital TVs have many constraints [6], when compared to PC. For example: hardware constraints (processing power and screen size); user input, (buttons, voice, touch screen and accelerometers); and different operating systems, like Android, Blackberry OS, iPhone OS, Symbian and Windows Mobile. These different characteristics must be taken into account when developing content for this kind of device [5]. Hence, the design and adaptation of content for all these platforms and devices is a tedious task. The presented architecture provides a Content Management System Adapter Architecture, which can be used to adapt all the content for mobile devices.

In general, the content adaptation envelops not only the adaptation of format and types, but also different styles, dimensions, data compression and specifications [4] [3], since quality of experience of the user can suffer from a not adapted (or poor adapted) media [1]. Also, different contents and information can be available for specific devices or systems, requiring a custom adaptation or generation of the content, which the CMS Adapter Architecture (the mCMS) can provide.

Most of the World-Wide Web content is manageable though CMSs (content management system). The CMSs provides a better content organization, increased access to resources and a greater organizational effectiveness as some of its' many advantages. CMSs can be used, natively or with plugins, for others devices, like TVs and Mobile devices, which can require the input of new contents. The mCMS provides an adaptation of the CMSs web content for the different device with the use of templates, providing a generic non-intrusive content adaptation. Also, natively, CMSs does not provide cache mechanism, compression of content, and version control (for the content on the client) like the mCMS

provides.

The mCMS was developed to be used together with popular CMSs like JOOMLA [7] and DRUPAL [2], and also with proprietary CMSs, CRMs (Customer relationship management), ERPs (Enterprise resource planning), and legacy systems. This work shows a test case of the architecture with a mobile application accessing the adapted content provided by the mCMS, which were created from resources made for web sites provided by a proprietary CRM and a JOOMLA CMS.

Mobile devices, normally, do not have connectivity all the time, needing some offline data in order to function when there is no network available. Also, the connectivity can still be very slow and expensive, depending on the network connection and the network carrier, requiring some form of data compression and caches techniques to reduce the data transfers. The mCMS provides a cache mechanism in order to fulfill these requirements. A version control of the content is also provided, so that the content transferred between the server and the client is only the difference between the data that is on the device and the new data from the server.

The mCMS consists in two parts, a server side and a thin client. The server is responsible for adapting the content from different sources throught the uses of templates, compressing the adapted content in order to delivery to the device and controlling the different contents versions. The thin client is an native application installed on the device, which is able to connect to mCMS Server, check for content updates, download the content package and save it locally on the mobile device. Also, using mobile devices can provide lots of features such as camera, navigation (location based features), social networks integration, statistics data, and much more. The thin client is designed to be a hybrid application, based on web patterns (HTML5, Cascading Style Sheets and Javascript) to provide the graphics interface, and a native code used to access the available devices capabilities.

This work is divided as follows, section 2 presents the mCMS architecture and thin client. Section 3 shows the test case in order to validate the presented architecture and finally section 4 presents the conclusions and future works.

2. THE MCMS SERVER

This architecture is build as a web service that adapts and creates content from CMS based on templates. An overview of the mCMS server can be seen on figure 1.

In this figure, the main modules of the architecture can be seen. The organization module is shown, that is composed by the resources provides, which can be the CMSs or even an web portals. These content are gathered by the controller and saved on a database. The controller is the service that connects all the different ones, and make them work together. Afterwards, the resources are adapted though the use of templates and them saved on a proprietary server or on a cloud server, like the amazon S3 and cloud front. Also, resources that come from secure services can be delivered and adapted, using the identity server. The Push notification services that are normally on the brand server, are accessed by a web service, and triggered by the controller

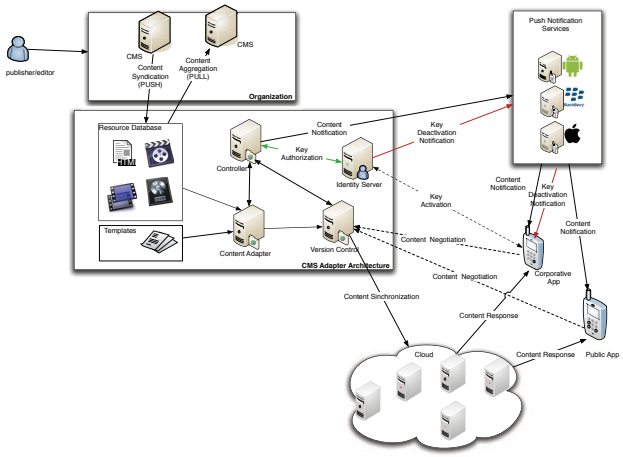


Figure 1: Overview of the mCMS server.

when new content arrives. This process is better illustrated on the Figure 2.

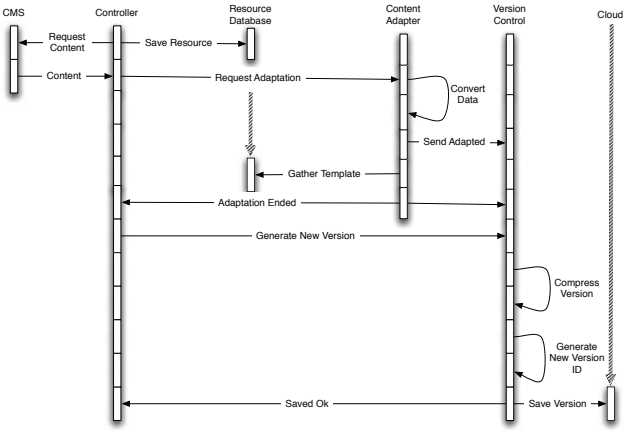


Figure 2: mCMs Execution Flow.

The templates for customization and configuration of the mCMS have to be registered in the controller by a developer/publisher/editor. These templates are built upon an XML based language, and they are the baseline of how the mCMS adapts the contents. Some contents can be specific for some device, like icons and logos, in this case they can be placed on the CMS server or inside the mCMS.

The architecture is built upon components. Based on experiments using this architecture in different application domains, a variety of technical implementations were used to abstract and create a general framework which different modules or plugins can be achieved for context adaption, personalization and contextualization.

The Identity Service is based on an AAA pattern (Authentication, Authorization and Accounting). It is responsible for generate certificates and key tokens used to implement a security connection between devices and the mCMS Server. Using this approach, we are able to guarantee access to special encrypted content and log all communication between

source and client. An extended feature of this enabling component is gather all statistic data available on the server and client sides an compile them on a report database.

The content version control is responsible for keeping each version of the adapted content. Every time the CMS Adapter Architecture gathers content from the source or when templates has changed, it generates a new version on its repository. This enabling component is built using a balk design pattern that only executes an action on an object when the object is on a particular state. In this case it only updates and generates a new version of the content, when all the content gathering and adaptation has finished doing its work.

2.1 Content Adaptation

The content adaptation is responsible for gathering the content from the CMS and adapting it to the required devices. This component normally uses a XML for the configuration of the adapter, and a series of XMLs describing how each of the content should be adapted. The process is simple, it uses a XML to gather information about the CMS service that provides the content, and how it should gather and adapt these content.

The component can gather the information in two ways, by push a notification generated by the CMS service, or in an automatic way, pooling from time to time (scheduler). The push notification requires that this service is implemented in the CMS, but it has the advantage that the content can be adapted as soon as published. In the automatic manner, this implementation is not required, but the content is only published for the media when the service runs.

This enabling component can support different content types and formats, like sound, music, documents, video and HTMLs. The adaptations of these content are made by templates, which are described by a XML document. These templates are implemented using Abstract Factory pattern, which defines the methods available for the content adaptation. The XML document describes how the component gathers the information, and which transformation need to be made for each platform.

The image adaptation is normally done using ImageMagick library, which is an free open source library, that the mCMS uses to change the image properties, like dimension, format and qualities. For the adaptation of audio or video, the architecture uses the FFMEPG library, which is an open source library, in order to convert the video/audio files, this way the adapter can change its properties, like the frame rate, format type, quality and dimensions. The configuration for each common device (mobile device) are registered in the application.

2.2 Thin Client

The thin client is responsible for: gather the data from the mCMS server, provide the server with the characteristics of the device, implement a cache system for the gathered data for offline use and gather of user data for statistics reports.

The client is developed as a framework, in order to be reused in others projects. It is mainly developed in object-C and Java, in order to work with the apple IOS and Google Android platforms. Others platforms are being developed, like for Window mobile, Blackberries and J2ME. This client can show different resources like, HTMLs, images, audio and video. It also can process and show a specify data structure for maps, which are used in the validation. Also, It implements components for localization system, cache management, statistics data gathering and integration with social network, like Facebook, Twitter and Google+.

This client was implemented mainly as a hybrid application, which is a mix of web app with native apps. Web apps are web sites, which all the content and logic are made exclusively for each device, but it cannot access some of the capabilities of the device. Native apps are implemented on the device, with a higher cost of implementation, but it can access all the functionalities of the devices that can be used by developers. Hybrid applications are a mix of both web and native apps, it uses mostly web for the interface, but it is still a native app, so it can use all the resources of the device, similar to the native app. Mostly of the customization of the application is done by XMLs.

The client uses extensively the cache for its contents. It uses this cache for providing the end-user with offline content and also minimize the data transfer between updates. This cache can also be cryptographic or compressed if needed by the application. Also, the cache consistency can be a problem in some devices, since the network data exchange can fail. This client only updates its version after verifying its hash code between the file that should be downloaded and the last updeted version. This client also uses a service to save the sessions' data, in order to gather statistics of the application use, if needed by the server.

In order to update the data in the client, the following workflow is used: first the device, if registered gets and push notification of new content availability; then when the user opens the application, the device authenticates and it gathers from the server the needed resources for its update; in the case the user has not an available connection, the application starts without this update process, by accessing the cached resources.

3. VALIDATION

In order to validate the architecture, the mCMS was used in a commercial project for Shoppings Centers. The application has all its data and contents gathered from different CMSs. This application aims to provide a channel of communication with relevant information about the mall, like the news, the sales, the attractions and also a map system for location of stores and parking.

One of the main problems of this kind application, without the use of the mCMS, is that the content comes from different services and different formats. With that, the mCMS need to have different configuration for gathering the content, but it can still show the same result for the different content providers. The CMSs used in this application comes from two different kinds, a Joomla! CMSs and a proprietary CRM. These services were already being used for the mall web sites, so the same content that was used for the sites is also used for the mobile application. All the content is gathered as pull service, running five times a day. The use

of mCMS architecture, has save some rework that needed to be done when customizing difference platforms for releasing different contents.

The server was developed in PHP and is composed of a linux with apache and MySQL. The thin client used for this application was adapted in order to work on iPhones and iPads and it was developed natively for these devices using the Cocoa framework. This client show the following contents: images, htmls, videos, maps and integration with social media (from Facebook and Twitter). This application also gathers some statistics data that is sent from time to time to the server to analyze the use of the software by the user. These information can be used latter for delivery of promotion, sales and events of the malls. One example of the mCMS that come from a JOOMLA CMSs that also provides the content for the web site can be seen on figure 3.

4. CONCLUSIONS

The devices nowadays have many difference characteristics and constraints, requiring specific content. With that, the devices need the publishing of specify content of the CMS or the adaptation of already published content. This work has presented a new adapter for CMS content adaptation, that provides a layer between the CMS and the devices.

Moreover, devices can have connection constraints, like availability and also this data can be very expensive. This presented architecture also provide an version control system and a cache system in order to provide offline data and also keep the data communication to a minimum.

5. REFERENCES

[1] F. Agboma and A. Liotta. Quality of experience management in mobile content delivery systems. *Telecommunication Systems*, 49(1):85–98, 2010.

[2] D. Buytaert. Drupa cms, June 2012.

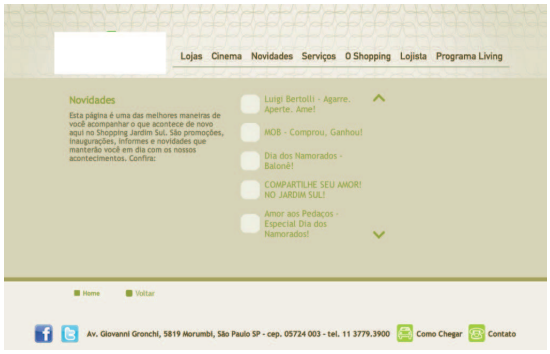
[3] D. Carvalho and F. Trinta. Content adaptation for multiplatform applications. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, WebMedia '09, pages 41:1–41:4, New York, NY, USA, 2009. ACM.

[4] T. Chaari, F. Laforest, and A. Celentano. Adaptation in Context-Aware Pervasive Information Systems: The SECAS Project. *Int. Journal on Pervasive Computing and Communications(IJPCC)*, 3(4):400–425, Dec. 2007.

[5] A. Hildebrand, T. C. Schmidt, and M. Engelhardt. Mobile elearning content on demand. *Information Sciences*, 5(2):94 – 103, 2007.

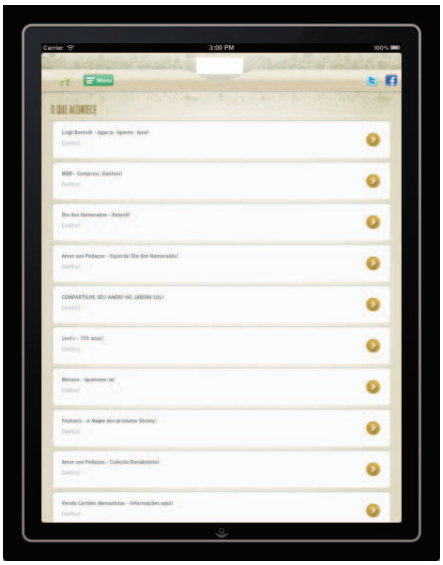
[6] M. Joselli and E. Clua. grmobile: A framework for touch and accelerometer gesture recognition for mobile games. In *Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment*, SBGAMES '09, pages 141–150, Washington, DC, USA, 2009. IEEE Computer Society.

[7] I. Open Source Matters. Joomla cms, June 2012.



(a) A list on the web site

(b) A list on a smartphone



(c) a list on a tablet

Figure 3: The mCMS in action adapting html content and image. The images were edited in order to remove the mark of the client (the white boxes).