On an Information Architecture for Mobile Applications

Sathiamoorthy Manoharan
Department of Computer Science
University of Auckland
New Zealand

***Abstract*—A number of websites are not easily viewable on modern mobile devices such as smart phones and tablets. To reach the audience one needs to architect information services so that the information can be rendered to suit the target device. This paper describes an information architecture suitable for delivering content to both native applications as well as browser-based mobile web applications. A case study based on a University environment is presented as an evaluation.**

***Keywords*-Mobile applications; information organization; information architecture; browser-based mobile applications.**

I. INTRODUCTION

Most websites that view well on a large desktop or laptop screen do not view well on the browsers of small-screen mobile devices. Some content providers such as BBC therefore provide a mobile version of their site when they detect that the requester is a mobile device. Some content providers also provide a native application that takes into account the user-interface paradigm of the device, and thus enabling a much better user experience on the device.

In this paper, we investigate the pros and cons of native applications and browser-based mobile applications, and discuss the requirements of an information architecture to suit both application types. We also present an experimental case study of developing a native application for use within a University environment.

The rest of the paper is organized as follows. Section II reviews some of the related work. The related work includes device capability recognition, content adaptation, and caching. Section III compares native applications to browser-based mobile applications. It also discusses a content delivery system built upon several of the ideas arising from the related work presented in section II. Section IV presents some requirements for an information architecture for content organization. Section V illustrates an experimental case study of developing a native application for use within a University environment. The final section concludes the paper with a summary.

II. RELATED WORK

A. Adapting Web Content to Mobile Devices

Some of the early work was to adapt the desktop web content to mobile devices either manually or automatically. Early mobile devices only had WAP access [1] rather than HTTP access, and supported only WML [2]. Oliveira and Camarao describe an early system that adapts HTML content for delivery to mobile devices [3]. The system, implemented in Haskell, converted HTML to WML so that the mobile micro-browsers were able to render the content. Google implemented a similar but a more sophisticated system that integrated into their search engine [4].

Mobile devices have a limited screen size. Consequently, modern mobile devices allow large content to be resized on-device to fit their screen, and allow zooming and panning the content so that areas of interest can be examined. While such zoom and pan access is fine for an occasional use, continuous browsing with zoom and pan can be tiresome.

Xie et al. describe how large pictures can be intelligently adapted to suit small screens [5]. This approach essentially identifies the regions of interests in the source picture so that these regions can be tailored to the target device. This is in contrast to the simplistic approach of re-sizing the pictures.

Lum and Lau present a content adaptation system that breaks large content into small coherent pieces tied together by a relationship [6]. In a broad sense, this is somewhat similar to the approach Xie et al. take in the context of pictures [5]. Lum and Lau consider textual content only.

The converted or adapted content will usually have temporal locality, meaning that recently delivered content may be re-delivered to other clients. Thus a suitable caching system to retain converted content over a period of time is required. Techniques from web caching can be employed in the mobile context [7]. Both textual and media contents benefit from caching upon conversion. Kara and Edwards describe such a caching architecture for pre-stored videos [8]. The system can equally be used for other type of content.

B. Device Capabilities for Content Adaptation

The Open Mobil Alliance (formerly the WAP forum) proposed user-agent profiles (UAProf in short) to tackle the explosive growth of a variety of mobile devices [9]. These profiles, based on the Composite Capability/Preference Profiles [10], contain the information that describe the capabilities of the devices. The profiles are simply XML files. HTTP requests from mobile clients contain an HTTP header, called *X-Wap-Profile* (or in older systems *Profile*), that points to the location of the profile. A server serving these requests therefore is able to consult the profile for any device-specific information. See Figure 1 that shows a set of headers including the *X-Wap-Profile* from a typical mobile device.

 ISBN: 978-1-61208-171-7

```
User-Agent: SonyEricssonP990i/R100 Mozilla/4.0
    (compatible; MSIE 6.0; Symbian OS; 306) Opera 8.60
Accept: text/html, application/xml, application/xhtml+xml,
    multipart/mixed, image/png, image/jpeg, image/gif,
    image/x-xbitmap, */*, text/x-vcard, text/x-vcalendar,
    image/vnd.wap.wbmp
Accept-Charset: windows-1252, utf-8, utf-16,
    iso-8859-1;q=0.6, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Pragma: no-cache
X-Wap-Profile:
    "http://wap.sonyericsson.com/UAProf/P990iR100.xml"
Content-Length: 0
X-Nokia-CONNECTION_MODE: TCP
X-Nokia-BEARER: GPRS
X-Nokia-gateway-id: NWG/4.1/Build89
Via: WTP/1.1 Vodafone wap2FTC
    (Nokia WAP Gateway 4.1/CD13/4.1.89),
    1.1 vlsp1:9010 (squid/2.5.STABLE10)
```

Fig. 1. A sample HTTP request from a mobile device. The header for user-agent profile, *X-Wap-Profile*, is shown emphasized.

Older devices may not have any profile information. Besides, there can be other issues with profiles: the profiles may be erroneous, may not conform to the schema, or may simply be absent at the location pointed to by the HTTP header.

For these reasons, some systems use an internal repository of device capabilities. Microsoft's ASP.NET mobile controls (formerly the Microsoft Mobile Internet Toolkit) was one such system which classified different devices based on the user-agent string in the HTTP request [11]. WURFL (wireless uniform resource file) is an open source profile repository which encompasses known profiles [12]. WURFL provides programmatic access to the repository for various languages (including Java, PHP, and .NET).

While such systems do not rely directly on the presence of the profile information in the HTTP header, they can become out of date very quickly. For instance, the capabilities of a newly-released mobile device may not exist in the repository until an update to the repository. Thus a repository based on dynamically caching user agent profiles is useful [13].

III. Native Applications vs. Browser-based Mobile Web Applications

Modern devices converge in terms of capabilities. Especially in the high-end or smartphone market, the devices have similar screen sizes and comparable resources and capabilities. With this in mind, some content providers (such as BBC) provide mobile sites targeted to this generic class of devices. In addition, some content providers (such as BBC and New Zealand Herald) provide native mobile applications that take into account specific hardware or software features of the device.

Native applications are device and/or operating-system specific, and thus several editions of these applications need to be developed. However, a native application can exploit the hardware and software features of the device to present a user with a much richer experience than a comparable web application. For instance, GPS capabilities of the device can be used to integrate location-based services: selecting an address may reveal the address on a map. Similarly, telephony services can be integrated: selecting a phone number may prompt a phone call.

A native application needs to be installed on the device. This can result in an application overload on the device. If a particular application or site is well-used by the user, then it is worth the user's while to install a native application; otherwise a web application can be a better choice since a web application is run through a browser.

For this reason, there is a case for developing both a web application and a native application. For example, a student or staff at a University may install a native application for the University on the device; while a casual visitor to the University may be better off using the University's web application (or site).

Thus both of the following are desirable:

1) browser-based mobile web applications adapting content using device capabilities, and
2) device-specific native applications offering rich user experience.

An architecture of a system capable of serving content for both browser-based web applications and native applications is illustrated in Figure 2.

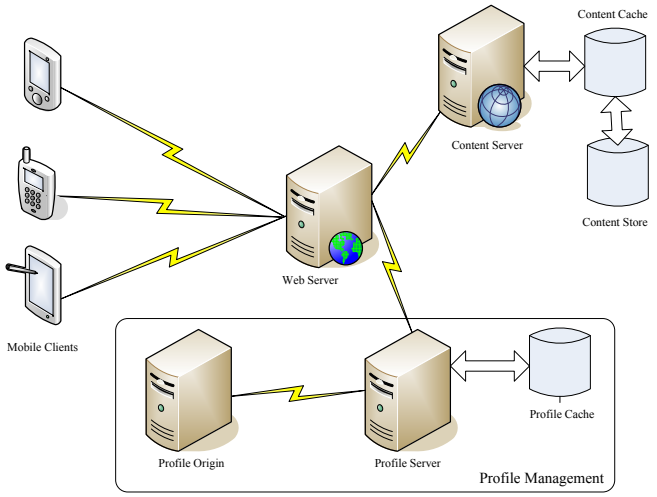


Fig. 2. Architecture of a content distribution system.

The workflow in the architecture is as follows. The *web server* receives the request for content from the mobile client (either directly or through a WAP gateway).

For a browser-based application, the server examines the header to get the profile location. If there is a profile location present, then it passes this to the *profile server* and acquires the profile. If there is no profile location present, it passes the user agent string to the profile server, and gets a default profile based on the user agent string. The profile and the request are then passed to the *content server*. The content server forms

 ISBN: 978-1-61208-171-7

content tailored to the profile, keeps a copy of the tailored content in the content cache for future re-use, and passes it along to the web server. The web server then delivers the tailored content as the response to the HTTP request.

For a native application, the web server passes un-tailored content, sourced from the content server, over to the mobile client where the content will be adapted to suit the device. Profile management is not required for native applications.

IV. Content Organization: Requirements for an Information Architecture

Organizing content to suit intended delivery is a data design task. This is specific to the audience of the content.

For a news organization (such as BBC and New Zealand Herald), the main content is news items. Often, a picture is associated with a news item. Such a picture can be used either as an icon or to make an otherwise textual reading interesting. A news item is categorized. Popular categorizations include *National*, *World*, *Sports*, *Business*, and *Technology*. A news item may fall into more than one category: for instance *Business* and *Technology*.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="Courses" xmlns=""
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="Courses" type="CoursesType"/>
   <xs:complexType name="CoursesType">
      <xs:sequence>
         <xs:element name="Course" type="CourseType"
         minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="CourseType">
      <xs:sequence>
         <xs:element name="Title" type="xs:string"
            minOccurs="1" maxOccurs="1"/>
         <xs:element name="Code" type="xs:string"
            minOccurs="1" maxOccurs="1" />
         <xs:element name="Semester" type="xs:string"
            minOccurs="1" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

Fig. 3. A sample XML schema for describing a list of courses

For a University, the main content for a student-oriented application is a list of programmes and courses (see Figure 3 that shows a sample XML schema describing a list of courses). In addition, there will be other items such as staff contact details and current news from the University. If the application is intended for a postgraduate student or a staff member, then the requirements will be quite different.

A user expects to see consistency in the look of an application, whether browser-based or native. To achieve this, it is important to have consistency across the content organization. For instance, if phone numbers are stored, all numbers should have a consistent format (e.g., +33 4 1234567). The database schemas should reflect such consistency.

V. A Case Study: A University Application

For evaluation purposes, we constructed a native mobile application for a University department. The application's intended audience is the undergraduate students in the department.

The information provided by the department's website was rationalized in the context of a mobile application. For example, large pictures are not useful in a mobile application. Course information provided by the department along with the contact details of the teaching and support staff were deemed to be the most useful to the students. The department also provides an RSS (Really Simple Syndication) news feed, and this feed was picked up as a showcase. Some images depicting the current departmental research activities were chosen to decorate the application and to break the largely text-only feel.

The information is then populated consistently into a *Content Store*. A web service was set up to supply the various content on demand. Two forms of the service were set up: one a SOAP-based service [14] and the other a RESTful service [15].

RESTful services, when based on HTTP GET, naturally lend themselves to caching. They are also lean, not having the overhead of SOAP. Besides, not all platforms support SOAP-based services well. RESTful services, therefore, are an attractive alternative.

The appendix provides some screenshots of a native mobile application using the information services.

VI. Summary and Conclusion

It can be difficult to view a number of standard websites on modern mobile devices (such as smart phones and, to some extent, some tablets). This is because these sites do not take into account the limited screen real-estate on mobile devices. A native application on the device can virtually show the same information as a standard website, but can do so in a manner that fits tightly with the user-interface paradigm of the device, thus presenting a much richer user experience than a web page. This paper described an information architecture suitable for delivering content to both native applications as well as browser-based mobile web applications. A case study based on a University environment is also presented as an evaluation.

References

[1] WAP Forum, "Wireless application protocol architecture specification," WAP Forum, Tech. Rep. WAP-210, July 2001.

[2] ——, "Wireless markup language specification," WAP Forum, Tech. Rep. WAP-191, February 2000.

[3] P. I. Oliveira and C. Camarao, "Adapting web contents to WAP devices using Haskell," in *Proceedings of the XXI Internatinal Conference of the Chilean Computer Science Society*, Punta Arenas, November 2001, pp. 223–232.

[4] Google, "How does Google modify web pages for mobile viewing?" See http://www.google.com/wml. Last visited September 2011. [Online]. Available: http://www.google.com/support/webmasters/-bin/answer.py?answer=35312

[5] X. Xie *et al.*, "Browsing large pictures under limited display sizes," *IEEE Transactions on Multimedia*, vol. 8, no. 4, pp. 707–715, August 2006.

 ISBN: 978-1-61208-171-7

[6] W. Y. Lum and F. Lau, "Relationship-aware content adaptation of structured web documents for mobile computing," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems*, July 2005, pp. 168–174.

[7] D. Wessels, *Web Caching*. O'Reilly & Associates, Inc., 2001.

[8] H. Kara and C. Edwards, "A caching architecture for content delivery to mobile devices," in *Proceedings of the 29th Euromicro Conference*, September 2003, pp. 241–248.

[9] WAP Forum, "User agent profile specification," WAP Forum, Tech. Rep. WAP-248, October 2001.

[10] G. Klyne *et al.*, *Composite Capability/Preference Profiles: Structure and Vocabularies*, January 2004, W3C recommendation.

[11] P. Yao and D. Durant, "Microsoft mobile internet toolkit lets your web application target any device anywhere," *MSDN Magazine*, vol. 17, no. 6, June 2002.

[12] L. Passani *et al.*, "WURFL: Wireless universal resource file." [Online]. Available: http://wurfl.sourceforge.net/. Last visited September 2011.

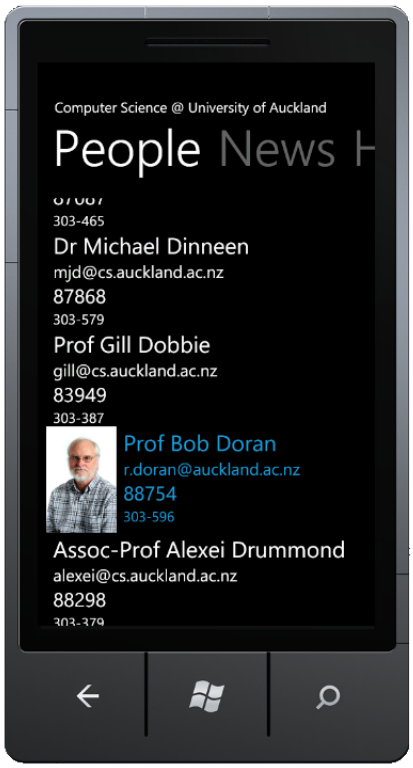
[13] S. Manoharan, "Dynamic content management and delivery for mobile devices," in *Proceedings of the International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. Papeete, French Polynesia: IEEE Computer Society, November 2007, pp. 63–67.

[14] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," *Internet Computing, IEEE*, vol. 6, no. 2, pp. 86 –93, Mar/Apr 2002.

[15] R. T. Fielding, "REST: architectural styles and the design of network-based software architectures," Doctoral dissertation, University of California, Irvine, 2000.

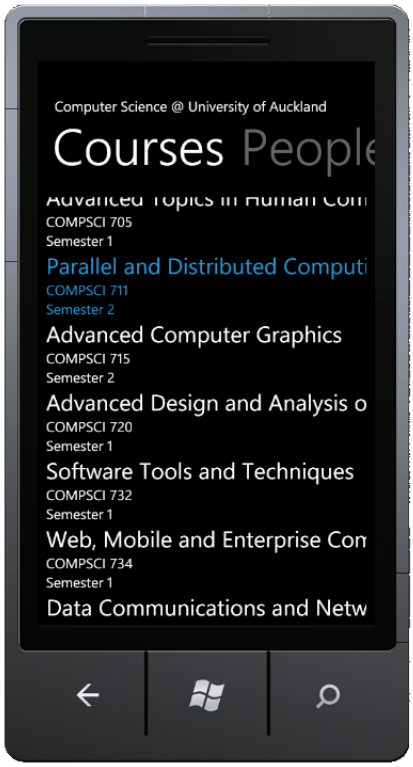
APPENDIX

This appendix illustrates some screenshots from the case study.



Selecting a person from the staff list shows a thumbnail picture of the selected staff member.

Similarly selecting an email address invokes the mail application on the device to compose an email to that address; and selecting a phone number prompts to dial the selected number.



 ISBN: 978-1-61208-171-7