“Adaptable web applications for Desktop and Mobile clients”

Student: Dr Black

Course: Internet Software Development Year 4 (BHWS4)
Supervisor: Dr. Richard Foley

Submitted for the Degree of BSc in Internet Software Development, 2005-2006

Except where explicitly stated all work in this report, including appendices, is my own:
Signature: _____ Date: _____

Abstract

Web applications can no longer rely on serving only desktop client browsers as their sole responsibility since mobile browser and hardware technology advances rapidly. Fifty million smartphones a year are shipping worldwide and this explosive growth of the mobile phone market fuels an increasing requirement for web applications that can cater for these devices. Since so many people will be using smartphones this poses interesting questions for reaching out to potential new users over the web. With the climate of web pages offering poor or little mobile web access integration; mobile websites do not offer comparable service to the desktop version which is an area that could be improved. In this paper the issues with HCI considerations are explored in a self adaptive mobile web application context. Exploring the uses of Java Model 2 design patterns, this paper shows how this paradigm can be used to create web applications that can adapt their content to suit both desktop and mobile devices automatically upon request.

Contents

1. Introduction

Web applications can no longer rely on serving only desktop client browsers as their sole responsibility since mobile browser and hardware technology advances rapidly. Research Company Gartner claim by the start of 2006 it is expected that fifty million smart phone handsets will be sold worldwide (BBC News, 2005). This explosive growth of the mobile phone market creates an increasing requirement for web applications that can cater for these devices. Since so many people will be using phones this poses an interesting area for businesses to literally reach out to new customers over the web. A recent Macromedia poll (Macromedia, 2005) suggests that "more than one quarter of U.S. adult cell phone owners polled (27%) claim that the information available from their cell phones is not useful for their day-to-day activities". This highlights that mobile websites do not offer comparable service to the desktop version which is an area that could be improved. It has been suggested that "since most Web contents are authored for a desktop environment, they are inadequate for mobile devices with limited display capabilities". (Kim & Lee, 2005)

The main problems associated with accessing the web with a mobile device is of a technical nature in relation to the capabilities of the specific device. Its physical size is the most apparent hurdle to delivering content as its screen size and resolution is limited immediately by this factor. With typical screen sizes for mobile phones being around the 180x220 pixel area on a 2 inch screen (Carphone Warehouse, 2006) this affords much less resolution on which to display content compared to a modern desktop computer at 1024x768 on a 17 inch screen. Similarly, the input capabilities are vastly different between the desktop interface and mobile interface. While it is almost unthinkable for a desktop computer accessing the World Wide Web to not have a keyboard and mouse interface, both of these basic interface elements are not available on the typical mobile phone. Filling in forms or entering large amounts of text may be unsuitable for mobile devices as keyboard capability exists with certain ergonomic constraints such as multiple button presses to select a certain letter, rendering existing text input forms cumbersome at best. This could pose additional problems when used in an internationalized application where special inflection characters are required in sentences so as not to completely distort their meaning. With existing web application development designed around traditional HCI guidelines for desktop computer browsers; this can cause difficulties for mobile browsers being able to display content in a meaningful way. It is relatively recently that even the biggest internet brand name companies have embraced the use of content adaptation methods for serving mobile devices in parallel with desktop devices. For example EBay – arguably one of the biggest internet based businesses in the world – only introduced a mobile targeted interface for their auctions site during the course of this research project (EBay Mobile, 2006). As more companies and researchers acknowledge the requirement for mobile websites that are comparable to desktop client targeted sites there have been many suggestions as to how this can be accomplished. Proprietary software applications, transcoding proxy techniques or by developing a sister mobile site are solutions to the problem that have been suggested for years; however they have not managed to break through into widespread commercial use. These techniques all work with varying degrees of success, but as mainly automated transformations never produce one hundred percent reliability which may be a factor that frightened off commercial interest. It is however in the author's opinion that a web application should be able to cater for both client types

internally without relying on a third party software product to add important functionality after market.

The focus of the project will be to investigate and apply programming methods to compare their effectiveness in implementing an adaptable site using java web technologies. This project aims to compare the capability of a simple web application to deliver a web site which conforms to a set of Human Computer Interface (HCI) guidelines researched for this project. The web application will be developed twice using Java Model 1 and Model 2 architecture on the Java 2 Standard Edition platform (Sun Microsystems, 2006). Comparing the two web applications will involve drawing up HCI criteria to contrast the delivered view and to show how effectively they deliver content to both mobile and desktop clients. Both websites should deliver acceptable content from the same data store dynamically to the client. The Model 2 application will take advantage of the paradigms attributes of content and presentation separation to deliver not only the dynamic content, but in addition a dynamic view tailored to the client accessing a page. The Model 2 application should base this decision upon the information that can be obtained from the client request header.

In summary, the project will involve the following key stages. Initially a literature review will be conducted to investigate the issues within creating and deploying mobile web applications. It will also be used to discover the HCI considerations of implementing both desktop and mobile web pages including issues with presentation, input and existing methods for adapting content. In addition, the technologies available to adapt content will be noted and adaptation methods implemented in those technologies will be evaluated. At the end of the literature review stage, an understanding of the current design challenges in relation to design of mobile web applications will be presented along with an appraisal of existing technologies for web application development in relation to the project research questions. The second stage of the project will involve a problem and systems analysis stage where the findings of the literature review will be used in order to design and create suitable test applications to address the projects hypotheses in relation to HCI constraints in desktop web sites viewed on a mobile browser. Here a set of technical and HCI guidelines will be derived with which the test phase of the project will be based upon. The thirds stage of the project will involve designing and implementing the two web applications. Drawing on the technical findings of the literature review along with the guidelines from the problem and system analysis stage, suitable test applications will be produced. Finally, in the testing and evaluation stage the HCI guidelines will be applied as a checklist and will contrast the resulting adherence of the web pages when displayed on desktop and mobile client browsers.

This project intends to address the following research question and hypothesis.

Can an adaptable web application serve both desktop and mobile browsers dynamically?

- Model 2 architecture will facilitate an application capable of serving customised views dynamically.

The results of this project would be of interest to anyone wishing to find out about the viability of developing adaptable web applications using the J2SE platform using established technologies.

2. Literature Review

In this section the various issues in the field of mobile web application development are discussed. In addition existing methods for solving content incompatibility problems will be compared and contrasted in the context of this project. Finally, it will discuss programming techniques available that could be used to answer this papers research questions.

2.1 What is a mobile web site?

A mobile web site differs from a traditional desktop targeted web site in a number of key ways. The primary concern for implementing a mobile website is that the capabilities of a mobile device will be significantly less than that of a desktop computer. Screen size, screen resolution, software functionality and input restrictions place a caveat on how a website can successfully display and remain interactive with the client (Freytag et. al., 1999; Kim et. al., 2005). These themes are echoed throughout research into the area of mobile website presentation since the emergence of web enabled mobile phone handsets and PDAs (Personal Digital Assistant). The issues generated throughout this period have been compiled into a World Wide Web Consortium working draft by the Mobile Web Best Practices Working Group categorising them succinctly within its requirements statement (Rabin et. al., 2006). This statement implies that for a web site to be able to fulfil its role in respect to presentation of data in a mobile context, its HCI considerations must be handled in a way suitable for mobile browsers/clients. It would appear this goal is at odds with developing a site for a desktop audience since far more client capability can be utilized there from data input, to animation or plug-in content like streaming video. In the author's opinion a site should adapt itself to the clients needs instead of catering to a lowest common denominator. This view seems shared further on in the working draft where they state if an adaptation process is used, then it should deliver content to make it more suitable for that device or to provide an enhanced user experience. If this was the case, it would imply that all web applications should provide some kind of adaptation method in which they can base their presentation on a client-by-client basis "information must be tailored to the constraints of mobile devices" (Billsus et. al., 2002).

2.2 Issues with mobile web applications

Designing web applications suitable for mobile applications can be tricky due do the obvious device constraints; however this would assume that any thought to a mobile client was considered to begin with. In this section problems associated with web site design in relation to mobile clients will be discussed separated in to software, ergonomic and hardware issues.

2.2.1 Presentation Issues

According to (Zhou et. al. 2006) "Most Internet sites aren't WAP-compliant" citing screen size as the major factor in this shortcoming, an opinion shared with (Kim et. al., 2005). Web pages on the whole are constructed poorly with regards to strict XHTML compliance with features emphasising upon the rich content features afforded by desktop browsers (Rabin et. al., 2006). Layout concerns are a primary factor as most modern websites are designed to be viewed on desktop screen resolutions without horizontal scrolling, commonly 800x600 pixels and upwards. A sample of common screen resolutions data mined from browsers accessing sites using

thecounter.com web counters has been summarised in Figure 1. This shows overwhelmingly that the typical website is being accessed by desktop browsers with a resolution of 1024x768 whereas only 2% of all page accessed could have potentially been made by a mobile device in the unknown category.

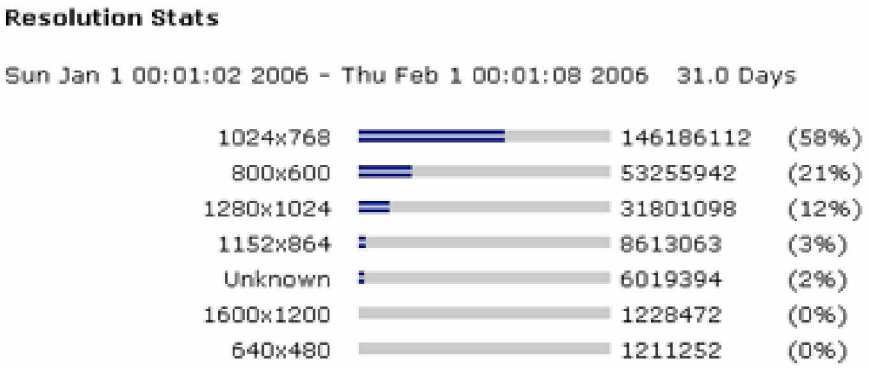


Figure 1 thecounter.com global stats for the month of January 2006 (thecounter.com, 2006).

On a small screen client such as a mobile phone, screen space is at a premium with resolutions far lower than the lowest available desktop resolutions – with reference to Figure 1, 94% of the desktop screen resolutions were over 640x480 pixels. Since the display of a page is the primary interface with the user it must be delivered in a legible format. If a web page is not presented in a suitable format to the client in regards to screen layout, it may result in an illegible page being displayed on a small screen client. As a result, this may introduce undesirable effects such as horizontal scrolling or misplaced objects (such as images) distorting the intended layout losing the context in which the information was intentionally presented (Kim et. al., 2005; Rabin et. al., 2006). Rabin highlights the importance of consistency in style within mobile website design to help the user create a mental map of the site. He also promotes that it is important if "considerable scrolling" is required to view the information displayed in a web page, the user does not receive any immediate feedback as to whether they have accessed the correct information they were looking for – especially if the top of the pages contain similar images or navigation links. This assertion is at odds with a study into the impact of small screen browsing of large screen targeted content, in which they found no conclusive evidence of degradation in usability in a handheld browser (Jones et. al., 1999). Although this study revealed no conclusive proof that small screens affected user efficiency, the author notes that the test subjects were 40 staff and students (half and half) from the Computing Science department with similar computer expertise. This result could have been skewed due to the technical expertise of the test candidates. Presumably all with minimal information technology skills, it is entirely likely they had more success than an untrained user would have had, since they most likely would have known to scroll the screen to find hidden user interface elements.

2.2.2 Input Issues

As a mobile phone client doesn't have the luxury of a keyboard and very few mobile browsers supplying a pointer/footprint for selection (Zhou et. al., 2006); input selections can be very limited. Rabin (2006) suggests that URIs (Uniform Resource Indicators) are "very hard to type" on a mobile due to the large amount of punctuation within a complete web address. Since a mobile phone handset has limited text input capability due to a numeric keypad the input of text in forms can also produce

headaches “Inputting a valid WAP URL through the existing numeric keyboard is tedious” (Kumar et. al., 2003). Although support varies from client to client, support for the ubiquitous “back” and “forward” buttons from desktop browsers may be lacking, so recovery from misclicks, missing links or error pages may be hindered (Rabin, 2006). It is the opinion of the author that the inclusion of navigation links should overcome this.

2.2.3 Device Limitation Issues

As mobile browsers often don’t support web plug-ins or scripting, content that relies upon these technologies must be adapted for a mobile context. This can result in unacceptable content types being returned which could render the display of a website illegible (Kim, et. al., 2005). “Display resolution must increase” according to (Zhou et. al., 2006), however that might not always be possible, or even desirable in the author’s opinion due to the limiting factor of handset size upon screen dimensions. This can be an issue that effects HCI considerations in a website which does not allow adaptation to a client browser type. For example if the site offered archived information but only in Portable Document Format (PDF), this would not be viewable at all on a mobile device unless it had support for a reader program which a desktop client could simply download with a plug-in.

Aside from the software related issues, hardware related issues also plague design considerations for mobile device capability. Content length is an important factor to be considered as memory is at a premium on a mobile phone compared to a desktop computer. In Appendix A, it is suggested in the guidelines to limit page content length to 20 kilobytes. This number is derived from the established limitations and specifications for current mobile browsers however, this can become a very tight window in which to work once images or any other binary downloads are included.

2.3 Content Adaptation Issues

The process of content adaptation can be described as varying the details of the mark-up, format of images, image size, colour to suit the characteristics of the client (Rabin et. al., 2006). This process of changing the content streams of a web site into a suitable format for presentation is specific to a client device. Generally this applies to adapting the desktop version of a website for use on a non-desktop client such as a mobile phone. Screen layout, scripting languages for user interface manipulation, graphics or multimedia elements may all be unsuitable in their default desktop targeted form, and must be adapted where possible for a mobile client. This makes the process of content adaptation extremely important if the message of a website is to be conveyed in a meaningful (and ultimately compatible) way to a mobile client device. Categorised as server side, in network and client side, content adaptation is implemented in at least one of the three stages (Freytag et. al., 1999; Rabin et. al., 2006).

2.3.1 Client-side Method

Client side adaptation refers to the onus of the content adaptation being performed on the client side with no help from the server application. This effectively is what occurs when any client device accesses a web site that is not intended for that device. The client accepts the content as is and tries to render it. Presentation technologies such as Cascading Style Sheets are an ideal implementation of client side adaptation; however they are not fully supported by compact hypertext mark-up language so are

not appropriate for mobile adaptation in anything but up to date mobile handsets. Content length in the context of mobile phone handset clients, is a factor against this type of adaptation because if the client cannot physically receive the content in the first place it has no hope of adapting it. This type of problem would arise in the limited Random Access Memory resources available being overflowed with incoming content data or the client processor not being sufficient for adaptation operations. (Freytag et. al. 1999; Rabin. Et. al., 2006; Kim et. al., 2005). This kind of adaptation is wholly unsuitable for this project as in its very nature; the source web application makes no effort to deliver its content in a suitable fashion to the client. This form of adaptation has its place, however this is more suited to applications within the desktop sphere alone for accessibility access using user defined cascading style sheets.

2.3.2 In-Network Method

In-network adaptation is when content is prepared in intermediate waypoints between the server and client. This encompasses a lot of different technologies from third party scraping proxies to compression applied by mobile networks. An example of this type of adaptation is in AdaWeb (Freytag et. al., 1999) which runs as a proxy server bridging a mobile user to the rest of the web. Client device context information is saved on the AdaWeb proxy server which then filters and adapts content into a suitable format for that client before passing the response back. AdaWeb was an early research attempt into the area of content adaptation for unprepared desktop targeted websites. This type of adaptation is still about today (XS4ALL, 2006) however is still offered as public beta service presumably due to it's quality of service being low, or it just not being effective from an HCI standpoint. This type of adaptation can also be seen in certain dual-proxy architectures such as WebBee (Upatkoon, 2005) which uses a proprietary client-side proxy application which can adapt content which has been scraped by a separate server side proxy. As an academic area of interest in-network adaptation does seem to be intriguing however little progress in the field seems to have been made over the past few years. In the author's opinion in-network approaches are unsuitable as they are 'after market' attempts to solve an HCI dilemma, rather than addressing the problem that the original site does not afford appropriate consideration for non-desktop clients.

2.3.3 Server-side Method

Server side adaptation occurs entirely on the server end, so the responsibility for delivering the appropriate content to a client is in the domain of the web server or web application itself. This generally involves traditional server processes coupled with serving cached content for specific device types (Gu et. al., 2004; Kaasinen et. al., 2000). Extensible mark-up language (XML) coupled with extensible style-sheet language transformations (XSLT) is one of the most common implementations where an XML data source has a suitable XSL style sheet applied to it and the resulting XML is returned to the client (MediaLab, 2004). Client-server content negotiation is a process which is completed on the server side by recognizing the client and serving the most appropriate pre-defined content available. This technique involves detecting the client's type, browser software or capabilities then selecting a pre-defined content template for that client type. Commercial implementations of this type of adaptation have appeared with device and client profiling, where users are data mined and served content specifically designed for their mobile device, a popular commercial implementation such as Volantis (Volantis, 2006). The primary point of contact for server side adaptation is the client hypertext transfer protocol (HTTP) request header.

In addition, there are several emerging technologies such as Composite Capability/Preference Profiles (CC/PP) which are an attempt by mobile manufacturers and software vendors to create a more standardised method for listing and detecting client capabilities (Klyne et al., 2004).

Several issues relate to the reliability of client capability detection by a server using HTTP headers for content negotiation. The importance of being able to detect client browser type or any other miscellaneous information provided in the request header is extremely important to the success of a server-side adaptation engine. Since the only point of contact without third party software is the HTTP request header only a small amount of useful data can be retrieved (Klyne et. al., 2004). This small amount of data must be data mined for all it can possibly reveal in order to get a precise as possible view at the client capabilities. If this is not handled correctly, the adaptation could return the incorrect or even completely incompatible content to the client. Detection of client type and/or capabilities is made using the http header information from the client http request as stated in w3c HTTP 1.0 specifications. Within the header information is contained several fields of interest.

- User-Agent
- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language

These headers contain information on the client making a request including browser name, version and operating system, file types accepted, character sets accepted, encoding schemes (such as g-zip compression) accepted and languages accepted. As there is no real standardization in the User-Agent field's text, special care must be taken in examining it as it could hold minimal information about the client, or instead may contain unexpectedly helpful information on a client-by-client basis. Commonly, it will hold browser name and version with operation system version. Utilizing the information within these header fields, the server can take that information and try match to content as best as possible to the client.

Since this was originally developed for desktop browsers a requesting mobile client cannot for instance specify a screen resolution or HTML (in) capability. A major stumbling block in standard request headers are that they are not standardised, which can mean that updating of device lists may be required often. The problem of disparate User-Agent fields has been addressed in the form of CC/PP. "A CC/PP profile is a description of device capabilities and user preferences. This is often referred to as a device's delivery context and can be used to guide the adaptation of content presented to that device." (Klyne et al., 2004) CC/PP is based on a Resource Description Format (Manola & Miller, 2004) which is an XML document used to describe a resource, such as a mobile phone. It allows many features to be exposed to the server so that a more informed choice of content can be made by the server in response to a request. In the author's opinion server-side implementations for content adaptation at source is the most efficient and reliable way with which to deliver content to mobile clients since they do not rely on third party software or automatic transcoding. Creating a web application that handles all client types internally should provide the best results as both the desktop and mobile views of the site are created exclusively for those purposes.

2.4 Web Application Paradigms

When creating web applications in a Java environment, there are two main design patterns for programming web applications. These paradigms for Java web applications will be the basis for the comparison section of the project and their uses will be discussed in more detail here. The Model 1 design paradigm is what would be considered the traditional approach to web application design which has been in use since most scripting languages for server side applications were created. Application content (data) and view (Hypertext Mark-up Language) are intermixed in the same code, so that essentially the web page they produce is created by running program code in place between HTML tags. The Model 2 paradigm approach (also known as Model View Controller (MVC)) is a newer design pattern where application content and view(s) are coded entirely separately, thus are not intertwined with each other like Model 1; reducing code clutter and increasing upgradeability and maintainability. Using both of these paradigms it should reveal a contrast between the traditional 'non-adaptive' website and a fully server-side adaptive web application.

2.4.1 'Model 1'

The Model 1 architecture can be best described as page-centric. The navigational map between pages is very linear much like static HTML web pages. When a request is received by the web server accessing a Model 1 Java Server Page (JSP), all responsibilities for that transaction are locked into that Servlet/JSP. It then executes as normal then it responds to the client directly (Figure 2). This particular Model does not scale upwards well to large complex applications (possibly running on Java Enterprise platform). Because there is only partial separation between business logic and presentation when using JavaBeans to abstract business logic, code redundancy creeps in when multiple views are presented in the one document. As JSP is interpreted by the server into a Servlet, this creates a significant amount of automatically generated code. This can cause problems when debugging as java exceptions will show significant volumes of low level code that was created by the Servlet container itself (Hall, 2002).

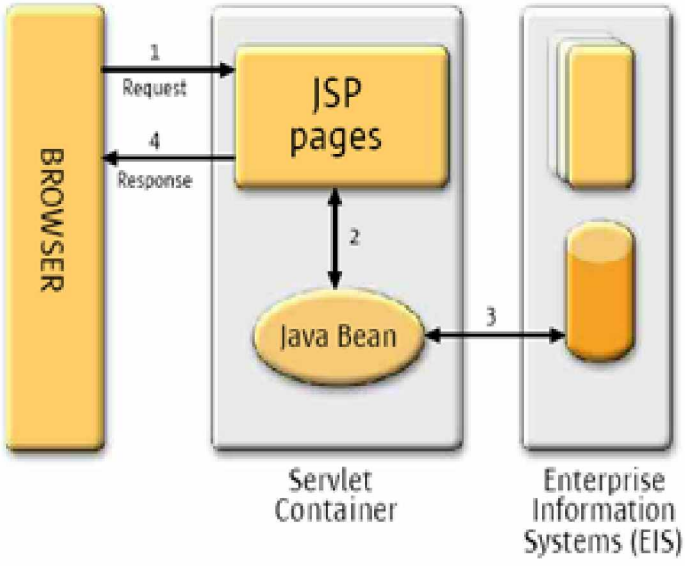


Figure 2 Model 1 design pattern (Shin, 2005)

2.4.2 'Model 2'

Model 2 architecture applications require a more complex setup than that of Model 1 application. When a request is received by the web server, the controller Servlet that was listening then performs business logic operations itself. Any results are saved into JavaBeans before selecting an appropriate view. This view is then populated with the JavaBeans and is then forwarded back to the client transparently (Figure 3). This approach scales gracefully to large and complex applications for several reasons. Code re-use of business logic is high since multiple views can be supported from a single controller. In addition presentation (view) and content (model) are completely separated. An additional bonus of this approach is it's maintainability since debugging is far more graceful as the controller and business logic are pure Java code, the only auto-generated code comes from simple JSPs for display only. This paradigm should provide an excellent foundation for a application that requires to have substantially different views, in this case a desktop version and a mobile version.

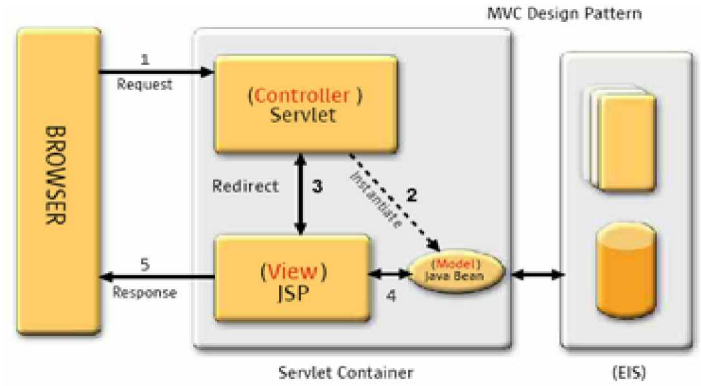


Figure 3 Model 2 design pattern (Shin, 2005)

2.5 Web application frameworks

There are a myriad of web application frameworks out there, however in order to maintain the schedule of this project, only the Java based solutions will be examined. All of the following frameworks will provide the support and essential functions to the development of the project applications.

2.5.1 Java Server Pages

JSP (Sun, 2005) provides a rapid facility for producing dynamic web pages that are platform independent. They allow scripting elements to be added into standard XML derived documents to generate content at runtime. When a JSP page is requested the server pre-processes it (which may result in a slightly longer execution time on first attempt), then executes the control code merging it back with static elements in the JSP to return the content (Hassan et. al., 2005). They are coupled with Java Servlet technology in that a compiled JSP page is turned into a Servlet by the web server providing an excellent implementation of the Model 1 paradigm. This should provide a sound basis for the non-adaptive web application backend code.

2.5.2 Macromedia ColdFusion

This is a similar technology to JSP but uses more HTML-like tags as its syntax. Current versions run off a Java based platform (Macromedia, 2006). This is a commercial solution that is more 'out-of-the-box' than JSP sporting built in web server and database connectivity, however comes at a commercial price. A developer edition is available for free, however it limits numbers of concurrent connections so may be unsuitable for testing in this project. ColdFusion will not be suitable for this project since it does not lend itself to implementing the Model 2 paradigm without significant programming effort unlike JSP.

2.5.3 WebWork

This is a newer implementation of MVC providing a Java web-application framework. It is built specifically with developer productivity and code simplicity in mind, providing robust support for building reusable UI templates, such as form controls, UI themes, internationalization, dynamic form parameter mapping to JavaBeans with robust client and server side validation (WebWork, 2005). As one of the most easy to use implementations of the Model 2 paradigm it would have been suitable for this project; however it has recently been announced that WebWork is being merged with Apache Struts.

2.5.4 Apache Struts

Apache Struts (Apache, 2005) provides an open source framework for creating Model 2 applications for the web. It offers many common features with WebWork, however it is a more mature, stable framework with a larger community contributing to its support and development in comparison. It has recently been adapted to a sister version of struts called "shale" - which allows integration with Java Server Faces, a JSP technology from sun. The classic version of struts still exists and is fully supported, although it has been renamed to "action" to differentiate between the two.

2.6 Review Conclusions

The concept of content adaptation through a web proxy is a popular area for research. It facilitates the adaptation of in most instances, any website, for use with a mobile client. Although this sounds an attractive proposition, it is the author's belief that the popularity of this type of solution owes itself to a general lack of adherence to XHTML standards within desktop web applications. Automatic document transcoding results in practice are unpredictable, usually favouring the simple websites to adapt and stumbling over multiple tiers of nested HTML tables in a website that really needs adaptation (Hwang et. al., 2003). Manual transcoding techniques require a web author to manually select and check the adaptation of websites, however this relies on him picking the correct sites to adapt (e.g. write a WebBee scraping script for a shopping site). Should the script not exist for a certain site the user's browsing experience ends there.

"Servers need a simple and elegant solution for displaying content effectively on mobile devices" (Zhou et. al., 2006). A solution that realizes that should be possible to implement with adherence to the standards set out by the w3c in (Rabin, 2006). Also promoted is promotes the view that a desktop web application has to be able to provide a reasonable experience to a mobile client in regards to what is termed "Default delivery context" summarized in *Appendix A*.

Utilizing a simple website design model and following the strict XHTML schema, it is the author's opinion that a simple self-adapting web site could be created using existing web application frameworks. This website should be able to automatically cater for desktop and a selected mobile client type without need of an external proxy program for adaptation. Model 1 and Model 2 programming paradigms can deliver a single web application with multiple views (Hall, 2004). Comparing those directly against each other in solving the same problem should provide an insight into the merits of each design approach in relation to a strict code implementation.

3. Methods

The project compares two of the most popular programming models for Java web applications, model 1 and model 2. It is recommended that if an application is sufficiently complicated that it will require substantially different presentations, Model 2 architecture should be used in lieu of Model 1 (Hall, 2005). This echoes the suggestion that the web is moving more towards a Model View Controller (Model 2) based paradigm to cope with the wide range of client types appearing (Ponzo et al., 2004). As this project compares two of the main web application programming paradigms a choice to use either one is not necessary like creating a single web application. "The choice of a framework is done early in the life of a project and too often with little thought about the future impact of such choice" (Hassan & Holt, 2004).

3.1 Action Research

In order to address the research question posed in this project an *action research* method was selected as the direction in which to take the project forward. Action research involves working on a specific problem and evaluating the results. In this case, the project is to develop and test prototype web applications using specific programming paradigms and evaluate them in producing a mobile and desktop compatible web site. This type of research was chosen over alternatives like *case studies* and *surveys* because as a develop and test project in computing, the emphasis is on the creation of a piece of software and then evaluating it's effectiveness in solving a pre-defined problem. This goal mirrors the definition of action research itself. Case studies on the other hand tend to revolve around intricate studies of an existing system, where this clearly inappropriate for a prototype test project. Similarly, a survey based approach to research would be equally inappropriate as this is more suited to questionnaires involving user groups and again, more suited to evaluating an existing system.

Testing the viability of an adaptable web application using an action research method, the development of a prototype web application is most appropriate. The research methodology chosen for this project is *implement and test*, this is because it facilitates the creation aspect of the action research and provides a vehicle with which to evaluate against the research questions/hypothesis. Other research methodologies such as a *case study* would be unsuitable for this project as the test subject (the web application) does not exist to be studied yet. *Theoretical analysis* may have been useful, but does not fit in with the practice of action research.

3.2 Project Methodology

As an example of a real life website which has good adherence to desktop web design standards, the homepage of the Caledonian University website was used to illustrate some content for the prototype web applications (Caledonian University, 2006). The page itself is shown in its original form in Appendix B. Since the original university pages do not have any content adaptation facilities they form an ideal template to use for the display of information for in a desktop targeted context. Since the Model 2 application includes content adaptation, it also follows the usability heuristics identified to be compatible with a mobile device context as listed in Appendix A. In the context of the mobile adapted site, the functional and non-functional requirements of the Model 2 prototype are listed in Appendix C. These requirements were derived from the usability heuristics identified in Appendix A.

The requirements are typical of a real web site because they provide a solid guideline as to how an adaptive web application should present its content dynamically. Although the content is not as extensive as the whole University website, this is not necessary in order to demonstrate content negotiation using the Model 2 design paradigm. As the negotiation process is identical for all pages within the Model 2 web application, and the non-functional requirements are as well; it would be repetitive to create more than a couple of examples of dynamic adaptation.

3.3 Qualitative Research

In order to implement the project's prototype web applications, varying Java based technologies were used. The four main elements of the prototype implementation can be classed into web container, application framework, JSP and the Adaptation module. Here follows the qualitative research analysis of the construction of the web applications. These elements are combined to create both prototype web applications and develop them using NetBeans IDE, as this has adequate coding, debugging and compiling facilities with which the author is already familiar.

3.3.1 Web Container

The web container is tomcat web server which is built into the NetBeans integrated development environment (IDE). This server is started and stopped automatically by the IDE when a project is compiled and run. As a Java based web server it has built in support for Java Server Pages and provides common HTTP web server functions. It can be configured to run as a local web server upon which to run the web applications developed for this project. It was used for both Model 1 and Model 2 applications. The initial setup for both web application projects in NetBeans was relatively simple, by selecting File > New Project > Web > Web Application the basic directory structure and prerequisite setup files for a java web application were automatically created through the wizard.

3.3.2 Application Framework

The application framework used was Apache Action Struts which is an established and stable framework for creating Model 2 web applications. This framework provided predefined classes with which Model 2 web application Servlets can be quickly created and configured using Extensible Mark-up Language (XML) documents. Setup of the Struts framework is slightly awkward to start with. The struts Java Archive (JAR) must be put into the class-path of the web application that will be using it. It is important to have this on the class-path of the application only, and not the environment as this may interfere with server operations or the IDE itself. This is for the purpose of compiling the servlets based on struts classes successfully within the IDE. This was achieved by placing the struts JAR into the "lib" folder of the Model 2 application in the IDE. Once the struts JAR was in place, struts class based servlets could be created and compiled successfully. Some additional tweaking was now required to provide the correct setup of the actions (web pages) so that they could be viewed. In the struts-config.xml file – an XML document used to store struts specific setup information – within the <action-mappings> section actions had to be defined for each of the Model 2 sites pages. These <action> tags allow a URI to be defined for a page, an Action class servlet associated with that URI to process requests and importantly, one or more <forward>s. These action forwards allow the definition of multiple JSPs or other view servlets which can be directed to once the

controlling Action servlet is finished processing the request. An example code snippet of an action mapping for a Model2 page view is in Appendix E. When a struts action is invoked, the struts framework passed it four key variables - ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse – with which decisions can be made. The ActionMapping object contains redirecting information defined earlier in the struts-config.xml <forward>. ActionForm is ignored in this case as advanced form processing is not required. The third variable is very important as this allows the application access to the client's user-agent request header. As discussed earlier, this forms the basis of the content negotiation logic in the Adaptation module (discussed shortly in 3.3.4).

Lastly, before the mappings will work, the web.xml file for the web application required some additional lines in relation to the struts framework. A servlet mapping was added on Action servlets so they gain the extension ".do" in URLs to help differentiate struts actions servlets from normal Java servlets. This point would have been where to add custom tag library definitions into this setup file for good practice, however in a non-production setting including them directly in the JSPs is acceptable.

3.3.3 Java Server Pages

The page generation for the prototype web applications were implemented for both Model 1 and Model 2 approaches using Java Server Pages (JSP). These provide a more efficient way with which to present web content using Java than using Java Servlets directly. This is achieved by allowing blocks of Java code to be inserted within HTML rather than HTML being inserted in blocks of Java code – saving editing time and compiler issues. The Model 1 application simply was built within a JSP file with no external capabilities. The use of JSP for the Model 2 paradigm application was somewhat different. In the Model 2 implementation, the initial request to the homepage action goes through the action servlet first. This then redirected the request to an appropriate view based on the results of the Adaptation Module.

3.3.4 Adaptation Module

In addiction to the predefined technologies used to create and deliver the prototype web application pages, a 'Content Adaption module' was created for use within the Model 2 prototype Action Servlets. This module was developed as a Java Class with methods that would take a Java (HttpRequest) object then return a Boolean value as to whether the User-Agent string indicated whether it was a mobile browser or not. The adaptation module source is listed in Appendix D. This is a vital part of the Model 2 prototype as it facilitates detection of the client type and allows the Model 2 website to adapt itself to return an exclusive view designed for the mobile or the desktop client accessing it.

3.4 Expert Analysis

A qualitative research assessment of the construction and technical differences between the Model 1 and Model 2 web applications was conducted. This assessment compared the two web applications in the way in which they are constructed, how complex they were to create and their relative strengths and weaknesses at implementing the functional requirements. In addition, comparisons on the success of each model in test browsers were made. Using the Nokia Mobile Browser to emulate

a mobile device, the impact of each web application's functionality upon rendering content on a mobile client can be observed.

The analysis is valuable to the project in that the qualitative nature of data action research lends itself to can be appraised in an unquantifiable manner. This allows a more interactive experience and view to address the research questions and gain a greater understanding of the workings of the programming models.

In order to conduct the qualitative research of the effectiveness of the web applications in delivering content to different clients an expert evaluation was chosen as the vehicle for this. This will involve running both model prototype applications and 'visiting' them with a standard desktop browser (Firefox) and the Nokia Mobile Browser emulator.

4. Results

The results of comparing the two Java programming models at solving the adaptive website for mobile browser problems are broken down into categories derived from the findings of the literature review. This seems the most logical way to catalogue how each model compares under the identified trouble areas.

4.1 Presentation Issues

In the literature review it was discerned that presentation issues such as browser resolution and the need for horizontal scrolling in order to view a web page in context would be a major consideration as to the success of a particular implementation. Model 2 overcomes presentation issues by tailoring itself to the clients display limitations by not requiring the large horizontal resolution, and maintains it's compliance with the HCI heuristics in Appendix A. Images are also left untreated for mobile clients in the model 1 approach, which results in an overly large page footprint. They are also not tailored to have a web safe colour palette which causes problems in display.

4.2 Input Issues

Including navigation links and pagination into the Model 2 mobile version, allows it to overcome issues of misclicks and the lack of back/forward buttons in a desktop browser. The model 1 approach, maintains everything on a single page which in this case results in poor compliance with the HCI heuristics in Appendix A in regards to total page weight and excessive scrolling.

4.3 Device Limitation

Limitations in the device capabilities were identified in the literature as a cause for concern as large page weight (defined as download size) can have adverse effects on limited RAM resources of a mobile browser, or cause total collapse of the intended user interface. This was a problem in the untreated model 1 implementation as the download size of some of the images outstretched the recommended maximum download size in the heuristics in Appendix A. Model 2 on the other hand overcomes device limitation issues by keeping page length under 20k by service properly prepared images – both sized in regards to resolution and download footprint.

4.4 Model 1 vs Model 2 comparisons

The qualitative research into the construction and technical differences of the two prototype implementations revealed several points. In regards to the Model 1 approach, it was found that straight up presentation in model 1 relies heavily on client side adaptation for both client types. Since this is unreliable, the rendered content was inconsistent with that of the model 2 when serving the content to a mobile browser. As identified in section 2.3.1 there were incompatibilities with the layout itself, as well as the adaptation method used namely Cascading Style Sheets. The original page renders perfectly in Firefox and even displays excellent HCI practice with the use of liquid layouts to cater for different browser resolutions. However, as can be seen in Appendix B the rendered result in Firefox is hardly discernable when viewed in the Nokia Mobile Browser emulator. Without substantial redesign which would affect both client types, the model 1 delivery was unsatisfactory for mobile clients. Conversely, the Model 2 approach allowed the original design to be rendered to a desktop client browser as intended, but when visited with a mobile client performed

much better as the view served is coded separately and specifically with the capabilities of the mobile browser.

Model 1 from a construction standpoint is relatively simple compared to model 2. The model 1 implementation does allow for a much more rapid development than the model 2 as additional software (the struts framework) is not required, neither is supporting Java classes or the need to configure routing within the application in external XML files. The model 2 application required a lot of extra setup including not only adding servlet mappings to the application web.xml but also requires careful naming and linking of all action servlets and associated view JSPs. For an application as simple as the test prototypes here, it does seem that the model 2 approach is heavy handed, however its ability to easily handle the requirement for radically different views can be envisaged as incredibly powerful when used in a larger scale application.

5. Discussion and Conclusions

5.1 Resume

This project addressed the question *Can an adaptable web application serve both desktop and mobile browsers dynamically?* This question was attempted first by performing an intensive literature review in the subject area of mobile web applications and mobile web access. From the literature review it was identified that there were categories of issues in relation to mobile web access, design and performance. It was also discovered that a possible solution to these problems may exist in Java Model 2 design patterns for web applications. Leading on from this an action research approach was adopted to develop and test a prototype web application of that type and compare it qualitatively through expert analysis with a Model 1 designed benchmark. This evaluation was carried out and its results were summarized in section 4.

5.2 Conclusions

In conclusion to the findings of this project has the research question been addressed? If a Java based web application had to be developed that would dynamically adapt itself to the specific client needs then in the author's opinion the developer would have to seriously consider using a model 2 approach right from the start. From the suggestions made in the literature review right through to the evaluation, the model 2 paradigm has delivered a neat and powerful mechanism for the projects purpose. The prototype using the 'traditional' model 1 approach has floundered at basic point down to its lack of support for multiple views. Although the model 1 would perform far better in a simple application scenario due to its low maintenance and speedy deployment, in answer to the question posed in this project it does not fair as well as the model 2 paradigm.

Overwhelmingly the model 2 design paradigm has outclasses the model 1 approach taken in all aspects of the defined HCI heuristics, however had the investigation of this project been in a different direction (response times, technical software metrics) it may have had some more favourable results in regards to the model 1 approach.

5.3 Future Work

The project shows a favourable result in regard to model 2 paradigm development of Java web applications where different display heuristics are required on the fly. This would suggest that model 2 could also have some interesting uses outside the scope of mobile clients or perhaps be used as a 'swiss army knife' solution that could serve the same data but in different formats upon request from XML to ASCII art. A model 2 web application lends itself to be much more customizable without interfering with existing functionality that it could become a very interesting area to further research.

6. References

Apache Software Foundation, (2005) *Apache Struts Project*
http://www.apache.org/ Last Visited: 05-02-06

BBC News. (2005) *Mobiles head for sales milestone.*
http://news.bbc.co.uk/1/hi/technology/4697405.stm Last visited: 15-11-05.

Billsus, D., Brunk, C. A., Evans, C., Gladish, B., Pazzani, M., *Adaptive interfaces for ubiquitous Web access*
Communications of the ACM, **45**, Issue 5, (2002) 34-38

Carphone Warehouse (2006) *Search of 3G and smart phones only*
http://www.carphonewarehouse.com/ Last Visited 01-05-06

EBay Mobile (2006) *EBay Mobile*
http://pages.ebay.co.uk/mobile/ Last Visited: 07-05-06

Freytag, C., Neumann, L., (1999) *Resource adaptive WWW access for mobile applications*
Computer & Graphics **23** (1999) 841-848

Caledonian University (2006) *Welcome to Glasgow Caledonian University*
http://www.caledonian.ac.uk/ Last Visited: 04-05-06

Gu, W. & Helal, A., (2004) *An XML Based Solution to Delivering Adaptive Web Content for Mobile Clients*
Proceedings of the 2004 Symposium on Performance Evaluation of Computer Telecommunication Systems

Hall, M., (2002). In *More Servlets and JavaServer Pages.* Prentice Hall, New Jersey.

Hassan, A.E & Holt, R.C (2004) *A lightweight approach for migrating web frameworks*
Information and Software Technology **47** (2005) 521-532

Hwang, Y., Kim, J., Seo, E. *Structure-aware Web transcoding for mobile devices*
Internet Computing, IEEE **07**, Issue 5, (2003) 14-21

Kaasinen, E., Aaltonen, M., Kolari, J.,Melakoski, S., Laakko, T. (2000) *Two approaches to bringing Internet services to WAP devices*
Computer Networks **33** (2000) 231-246

Kim, H & Lee, K (2005) *Device-independent web browsing based on CC/PP and annotation*
Interacting with Computers **xx** (2005) 1-21

Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M.H., Tran, L., (2004) *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0*

http://www.w3.org/TR/CCPP-struct-vocab/ Last visited: 11-01-06

Kumar, V., Parimi, S., Agrawal, D.P., (2003) *WAP: present and future* Pervasive Computing, IEEE **02**, Issue 1, (2003) 79-83

Macromedia, Inc. (2006) *Use of mobile data services hampered by poor user experiences.* http://www.macromedia.com/macromedia/proom/pr/2005/mobile_survey.html Last visited: 03-11-05

Manola, F. & Miller, E., (2004) *RDF Prime* http://www.w3.org/TR/rdf-primer/ Last visited: 11-01-06

MediaLab, Telia Sonera, (2004) *Web Content Adaptation White Paper* http://www.medialab.sonera.fi/workspace/WebContentAdaptationWP.pdf Last visited 15-11-05

Rabin, J. & McCathieNevile, C. (2006) *Mobile Web Best Practices 1.0.* http://www.w3.org/TR/mobile-bp/ Last visited: 16-02-06

Shin, S. (2005) *MVC Pattern (MVC Framework)* http://www.javapassion.com/ Last visited: 12-10-05

Sun Microsystems, Inc. (2005) *JavaServer Pages Technology* http://java.sun.com/products/jsp/index.jsp

Thecounter.com (2006) *Global screen resolution statistics January 2006* http://www.thecounter.com/stats/2006/January/res.php Last visited: 09-02-06

Upatkoon, K., Wang, W., Jamin, S., "WebBee: An Architecture for Web Accessibility for Mobile Devices" Proc. of the 10th IFIP International Conference on Personal Wireless Communications, August 2005, Colmar, France.

Volantis (2006) *Intelligent Content Adaptation* http://www.volantis.com/ Last visited: 28-01-06

W3c, *HTTP/1.1:Request* http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5.3 Last Visited: 13-02-06

XS4ALL, *Experimental: Mobile Web Accelerator* (2006) http://www.xs4all.nl/uk/allediensten/experimenteel/mobilewebaccelerator.php Last Visited: 05-05-06

Zhou, B., Hui, S. C., Chang, K., (2006) *Enhancing Mobile Web Access Using Intelligent Recommendations* Intelligent Systems, IEEE **21**, Issue 1, (2006) 28-34

Appendix A

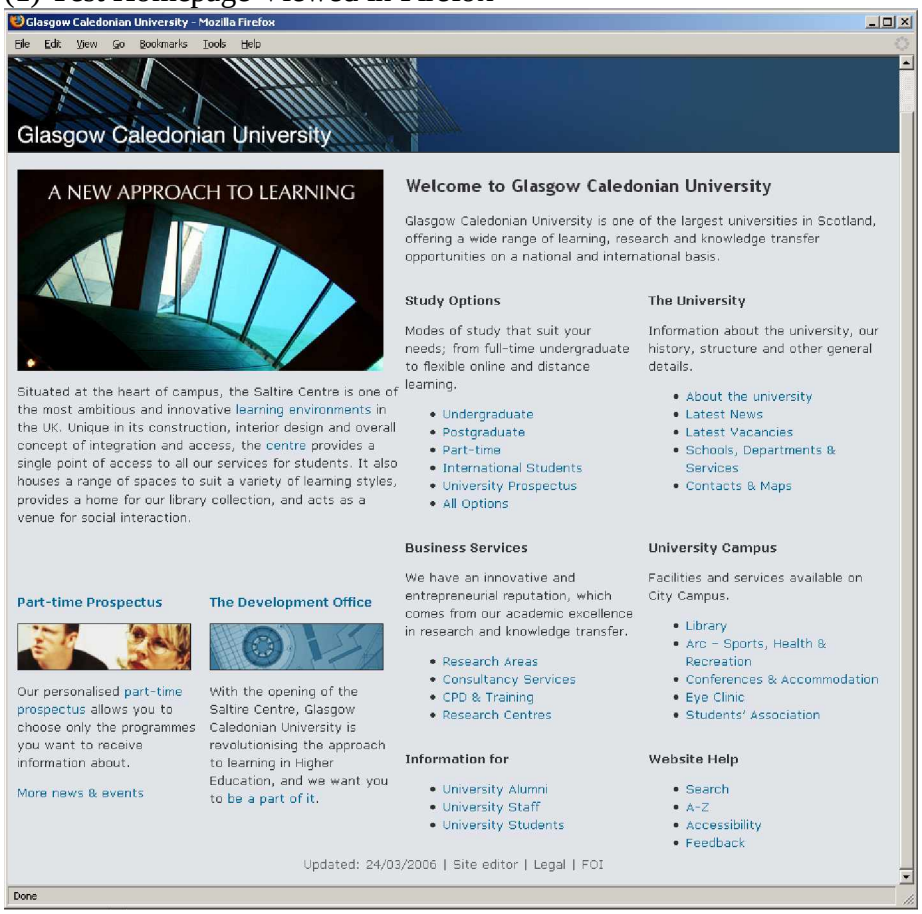
Usability Heuristics for a Mobile Web Application:

Usable Screen Width	120 Pixels, minimum
Mark-up Language Support	XHTML - Basic Profile [XHTML-Basic].
Character Encoding	UTF-8 [UTF-8]
Image Format Support	JPEG GIF 89a (non-interlaced, non-transparent, non-animated).
Maximum Total Page Weight	20 kilobytes.
Colours	Web safe. (A Web safe colour is one that has Red/Green/Blue components chosen only from the values 0, 51, 102, 153, 204, and 255.)
Style Sheet Support	External CSS Level 1 [CSS].
HTTP	HTTP/1.0 [HTTP1.0] or more recent [HTTP1.1].
Horizontal Scrolling	None or minimal.

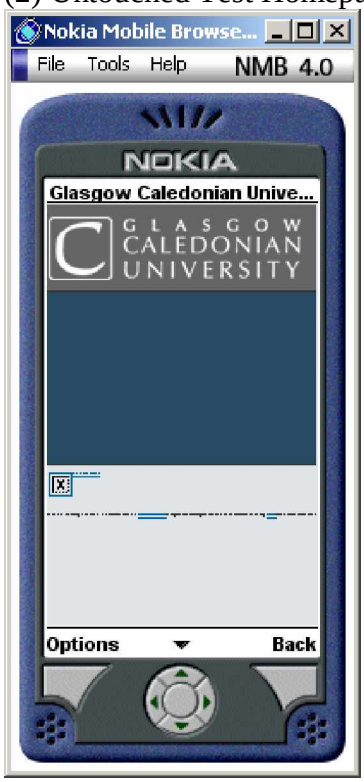
Adapted from (Rabin, 2006)

Appendix B

(1) Test Homepage Viewed in Firefox



(2) Untouched Test Homepage viewed with Nokia Mobile Browser



Appendix C

Functional Requirements of Model 2 prototype:

Functional requirements

1. provide 2 alternate presentations per page, one desktop, one mobile
2. must detect request headers on each request
3. each page must provide both views of itself dynamically
4. must test if client is desktop or mobile
5. must return correct content type

Non-functional requirements

1. both views use Apache Action Struts
2. desktop view must use XHTML-Transitional
3. mobile view must have 120px minimum screen width, maximum 220px
4. mobile view must use XHTML-Basic
5. mobile view must encode text using UTF-8 character set
6. mobile view may only use JPEG or basic GIF 89a images
7. mobile view total size (including images) must not exceed 20KB
8. mobile view must use web-safe colour palette
9. mobile view may only use external CSS1 stylesheets

Appendix D

Adaptation Module algorithm and source.

```
package project.modelTwo;

import javax.servlet.http.HttpServletRequest;

public class AdaptationModule {
     private HttpServletRequest r = null;
     private String u = null;

     public boolean isMobile( HttpServletRequest request
) {
          r = request;
          u = r.getHeader("User-Agent");

          if( u.indexOf("Firefox") != -1 )
               return false;
          else
               return true;
     }
}
```

Appendix E

An example action mapping for a homepage with 2 possible views in the Model 2 application

```
<action path="/modelTwo/getHomePage"
type="project.modelTwo.GetHomePageAction">
      <forward name="desktop" path="/WEB-INF/jsp/home_desktop.jsp" />
      <forward name="mobile" path="/WEB-INF/jsp/home_mobile.jsp" />
</action>
```