Xiaogeng Zhao

Supervisor: Professor, Peter G. Clayton

A Comparative Analysis of Java and .NET Mobile Development Environments For Supporting Mobile Services

Submitted in fulfilment of
the requirements for the degree of
Master of Science at Rhodes University

Computer Science Department, March 2003

RHODES UNIVERSITY
Where leaders learn

Acknowledgements

To my mentor and supervisor Professor Peter Clayton, for permanently keeping his door open to me, for always being willing to listen and help out in any way possible. The guidance received from him has been invaluable. But if I were to use only one word to describe my thoughts, it would be "trust". It is his trust in me that has made my research go smoothly; given me confidence; and above all it is this quality that makes him a respectable man!

To my parents, my dad Jianmin Zhao and my mom Danhua Jia: although we are ten thousand miles apart, my heart has not left you. Without your selfless support, I could not possibly have gotten to this stage in my life. It is your love that gives me a clear mind and helps me to conquer all difficulties; it is your love that makes you irreplaceable!

To my girlfriend Tammy for all the happiness and love she brings to me!

To my best friend Tim, Raymond and Ellen for all the good time we spent together!

To Helen for the proofreading!

To all the staff and my colleagues in our department for your constant support!

To my country for making me so proud!

This work was undertaken in the Distributed Multimedia Centre of Excellence (CoE) at Rhodes University, with financial support from Telkom, Lucent Technologies, Dimension Data, and THRIP. We also acknowledge the bursary support from National Research Foundation (NRF) and Microsoft.

Abstract

With the rapid development of wireless technologies, traditional mobile devices, such as pagers and cellular phones, have evolved from a purely communications and messaging-oriented medium to one that supports mobile data communication in general and acts as an application platform. As shown in a recent survey conducted by MDA[1], easy access to the present-day wireless Internet has resulted in mobile devices gaining more and more attention and popularity. The growth of and demand for mobile Web applications is expected to increase rapidly in the near future, as a range of software companies and mobile device manufacturers release increasingly accessible tools for creating mobile Web application and services.

From a variety of possible development environments of this kind, the author has selected and examined two leading contenders, the J2ME and the Microsoft .NET mobile Web application development environments. This document reports the product life cycle of pilot mobile web applications, designed and implemented in each host environment in turn.

A feature-by-feature investigation and comparison of the J2ME and .NET environments was carried out, covering the range of issues necessary for a complete mobile Web application development life cycle. The resulting analysis addresses features and efficiencies of the application development environment and the target deployment environment, the degree to which the resultant services are compatible on a variety of platforms, and the ease with which applications can be designed to be extensible.

The thesis offers an objective evaluation of the J2ME and the .NET mobile development environments, which highlights their strengths and weaknesses, and suggests guidelines for designing, creating, and deploying high quality mobile Web applications.

The research uncovers no clear winner across all categories assessed. J2ME currently favours situations in which bandwidth is limited and client side processing power is relatively sufficient, it exerts the processing power of mobile devices over distributed network environments. .NET requires a less constrained network throughput, but performs adequately on clients with more limited processing power, supports a more diverse target platform range, and offers a more efficient, in terms of development time, development environment. Both technologies are likely to receive significant user support for some time.

[1] Mobile Data Association (MDA) is the non-profit, global association for vendors and users of mobile data and their advisors. http://www.mda-mobiledata.org/

– "The White Rabbit put on his spectacles. 'Where shall I begin, please your Majesty?' he asked. 'Begin at the beginning,' the King said gravely, 'and go on till you come to the end: then stop."

Lewis Carroll, *Alice's Adventure in Wonderland*

List of Acronyms

ADO	ActiveX Data Object
ARPANet	Advanced Research Projects Agency Network
ASP	Active Server Page
AWT	Abstract Windows Toolkit
CDMA	Code Division Multiple Access
CDPD	Cellular Digital Packet Data
CGI	Common Gateway Interface
CHTML	Compact Hyper Text Markup Language
CLDC	Connected Limited Device Configuration
DARPA	Defence Advanced Research Projects Agency
EJB	Enterprise JavaBeans
GIS	Geographical Information System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GNU	GNU is Not Unix
GSM	Global System for Mobile Communication
HSCSD	High Speed Circuit Switched Data
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transport Protocol
IDE	Integrated Development Environment
J2EE	Java 2 Platform Enterprise Edition

J2ME	Java 2 Platform Mobile Edition
J2SE	Java 2 Platform Standard Edition
JAD	Java Application Descriptor
JAR	Java Archive
JCP	Java Community Process
JDBC	Java Database Connectivity
JSP	Java Server Page
JVM	Java Virtual Machine
MDA	Mobile Data Association
MIDP	Mobile Information Device Profile
MIT	Massachusetts Institute of Technology
OLE	Object Link and Embed
PDAP	Personal Digital Assistant Profile
RAD	Rapid Application Development
SDK	Software Development Kit
SMS	Short Message Service
SMTP	Simple Mail Transport Protocol
SQL	Structured Query Language
SSL	Security Sockets Layer
TDMA	Time Division Multiple Access
WAE	Wireless Application Environment
WAP	Wireless Access Protocol

WAR	Web component archive
WDP	Wireless Datagram Protocol
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
WWAN	Wireless Wide Area Network
XML	eXtensible Markup Language

Table of Contents

List of Tables

List of Figures

List of Code

Chapter 1: Introduction

– "There was a time when the fastest mode of business travel was by land or sea. Now it's by air. And there was a time when the fastest mode of business communication depended on wires and cables. Now it is wireless" [IBM, 2001].

This chapter covers the background, goals, and scope of this project. A general overview of the subsequent chapters is also provided.

1.1 Background introduction

1.1.1 The Internet evolution

"The Internet has revolutionized the computer and communications world like nothing before. The invention of the telegraph, telephone, radio, and computer set the stage for this unprecedented integration of capabilities. The Internet is at once a world-wide broadcasting capability, a mechanism for information dissemination, and a medium for collaboration and interaction between individuals and their computers without regard for geographic location." [Leiner, Barry]

The concept of the Internet was first introduced by J. C. R. Licklider of the Massachusetts Institute of Technology (MIT) in August 1962. He described this concept as a globally interconnected set of computers. Later on he headed up the research unit at the Defence Advanced Research Projects Agency (DARPA). In September 1969, as a result of the research, the Advanced Research Projects Agency Network (ARPANet) was born. Both government and academia joined the network rapidly so that by 1985 what is now known as the Internet had been established.

By October 1993 there were more than 800 Web servers in existence. By November 2002 the number of Web servers increased to more than "35,686,907"[Netcraft] with thousands of new Web servers joining every day. The Internet of the present day has become part of the daily life of individuals. It has become a universal applications and services platform for online and offline business, production management, national defence, education, entertainment, and so on. It is a strong factor with regard to economic growth. Some numbers show that there are "522 million users online world-wide. ICT Spending Continues to Grow from US$ 1.3 trillion in 1993 to over US$ 2.4 trillion (Digital Planet2002). Increases in E-commerce Market Share (Especially B2B):Internet Purchases in 2001 amounted to over US$ 600 billion -- US$ 516 Billion (B2B); and US$ 117 billion (B2C) (Digital Planet2002)" [Olive, David].

1.1.2 Mobile Internet and the wireless communication

As the Internet reaches a new level of infiltration, the fixed desktop connections along with the traditional online services become less and less suited to the rapidly changing environment and the increasingly demanding requirements of the Internet. This trend is likely to continue, as the entire IT industry continues along its speedy growth curve. A more flexible and powerful alternative Internet connection is needed, one that can retrieve the latest information quickly on a variety of devices that have a wide range of capabilities, without the constraints of geographic location.

Wireless technologies include cellular networks, wireless WANs, wireless LANs, metropolitan local-loop solutions, and so forth. They also include business considerations associated with the enterprise deployment and management of mobile computing devices.

The benefits of wireless networks are revolutionary, since connectivity no longer depends on a physical attachment. Local areas are measured not in feet or meters, but miles or kilometres bringing complete freedom and flexibility to users of the network. Deploying wireless networks can be more cost effective and more immediate than leased lines or trenching. From the perspective of data communication, wireless networking is able to support multi-directional relational database synchronization, two-way file transfer, and remote installation of applications. The emerging Internet-enabled telephonic devices, along with the increasing use of notebook computers and PDAs (Personal Digital Assistants), make the deployment and development of wireless solutions for business, technology and management easier than ever. People equipped with these Internet-enabled mobile devices, have access to their calendars, e-mail, and to data in general whenever they want regardless of where they are.

Apart from all the hardwire technologies that underpin mobile solutions, online mobile services are required which can intelligently provide information to a variety of devices in a pro-active manner. Nearly all major IT companies and mobile device manufactures are developing applications and services on various mobile platforms.

1.2 Project introduction

This project examines a variety of mobile Web application and service solutions developed by a wide range of companies. It focuses on the study, analysis, and comparison of two distinct new solutions: SUN Java mobile Web solution and Microsoft .NET mobile Web solution. With the backing of the two IT giants, SUN and Microsoft, these solutions are considered to have the most potential, and are likely to become major competitors in the future mobile application market.

1.2.1 Primary goal of the project

The primary goal of this project is to conduct a comparative study of the Java mobile Web solution and Microsoft .NET mobile technology under deployment conditions. The comparison focuses on the mobile Web application development life cycle with regards to different solutions, which include the mobile application development environment, deployment environment, application efficiency, application compatibility, application extensibility, and so on. The intention is to give guidelines on how to efficiently develop high quality mobile Web applications with high levels of portability and extensibility.

1.2.2 Experimental approach

This research takes an experimental system building approach to computer science, in which prototypes are developed to test the effectiveness of competing ideas and hypotheses. In this case, both the software engineering process and run-time target system were assessed from the construction of pilot applications.

Firstly, the Java mobile development environment was used to construct a relatively complex mobile Web application for Java-enabled devices, such as Java cell phones, and Compaq Pocket PCs (the "*Rhodes University Mobile Meal Booking System*").

Secondly, use was made of the Microsoft .NET mobile development environment to build a similar application with the same level of conceptual structure, but which targeted a different problem domain (the "*Rhodes University Mobile Geographic Information System*").

Then, based on the results and experiences obtained from both design and implementation exercises, a comparative analysis was made of the two development environments and their associated run-time support systems, and finally, conclusions were drawn and a set of overall guidelines for mobile application development drawn up.

1.3 Foundation

The discussion about Java and .NET mobile solutions proceeds as follows.

Chapter 2 introduces and compares a variety of mobile communication mechanisms, and mobile Web protocols.

Chapter 3 explores the development life cycle of a J2ME mobile Web application by going through the development detail of the *"Rhodes University Mobile Meal booking System"*.

Chapter 4 investigates the development life cycle of a .NET mobile Web application by examining the development detail of the *"Rhodes University Mobile Geographic Information System"*.

Chapter 5 conducts a comparative analysis of the development of J2ME mobile Web applications and .NET mobile Web applications and delivers guidelines with regard to the development of mobile applications.

Chapter 6 draws a conclusion for this project and also touches on the possibility of future extensions.

1.4 Chapter review

In this chapter, we first documented the background of this project, which includes a brief history of the Internet and an introduction to the wireless network. We then went through the research steps of the project, and laid out the goals to be achieved by the end of the project. We also briefly described the thesis and its layout.

Chapter 2: Wireless Communications and Protocols

– "The year 2003, 'E-mail everywhere' - e-mail becomes as commonplace and as necessary as the telephone. The year 2004, the Web (or a future version of it) becomes a generic 'front-end' for all information appliances, including cellular phones, TV sets, and telephones" [Leopold, Eileen].

Introductions and comparisons of several wireless communications and mobile Web protocols are given in this chapter, with the intention of providing an overall picture of the current situation in mobile Web technologies and their development prospects.

2.1 Categorizing wireless communications

Surprisingly wireless communications can even be traced back to 1920s when radio broadcasting was widely adopted. Unlike radio broadcasting, which sends the same message to multiple receiving devices in one direction, modern wireless communications technologies target the inter-communication between specific devices, such as paging and cellular phone communication. During the past few decades the growth of wireless communication has been explosive. At present there are more than 300 million wireless users world wide, and a wide range of different protocols and standards are available. To get a comprehensive overview and understand the available options in handling various data over wireless technologies, let us first take a look at some major wireless communication mechanisms that are in use today.

2.1.1 Paging communication

The first generation wireless person-to-person digital communication device is the pager. A pager simply emits beeps and/or displays a short text message to alert the receiver to phone the specified caller back.

Type of transmission:

To reduce the physical size and technical complexity there is one receiver inside every pager. No transmitter is included. Therefore the nature of pager communication is one-way. It is only capable of receiving and not sending data.

Speed and bandwidth:

The wireless networks that service most pagers are typically limited to 10 Kbps, which is only around one sixth of the speed of a standard 56Kbps dial-up modem.

Service range:

The paging service range is normally limited by the radio coverage of the paging stations.

Data processing capability:

Paging networks run on very low bandwidth. They normally have a very limited processing power and are equipped with a low-resolution screen. Due to these limitations pagers are only capable of processing and displaying text data.

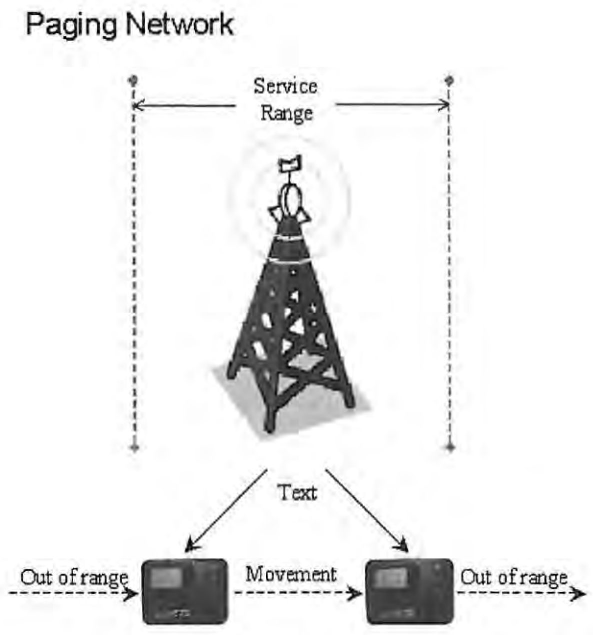


FIGURE 2.1.1-1 – WIRELESS PAGING NETWORK

Development and expectation:

Since the first commercial pager introduced by Motorola in the year 1974 the development of paging communication has been on the go for nearly thirty years. It achieved its success and dominated the wireless communication during the 80's and early 90's. However, with the tremendous impact of the cellular phone, next generation personal data communication devices, and their add-on services, such as the Short Message Service (SMS), currently, the development of paging communication is under great pressure. "Many industry analysts expect that over the next few years, paging-only devices will lose their market share to multifunctional devices, such as Internet-enabled mobile phones and newer electronic devices such as the Research in Motion (RIM) BlackBerry, which uses two-way paging to link users to their e-mail systems" [Wigley, Andy].

Paging Communication and mobile Web applications:

Traditional paging communication is one way and text only. It does not leave much space for interactive mobile Web applications, but with the emergence of two-way pagers and the wireless Internet access support, the pager is still an ideal platform for text-based mobile Web applications.

2.1.2 Cellular communication

Cellular communication is the technology that uses many base stations to divide a service area into multiple smaller regions (cells), and transfer calls from base station to base station using the cellular switching mechanism as users travel from cell to cell. With the cellular communication technology, for the first time, people can enjoy truly mobile communication. They can contact of one another no matter where they are and what they are doing, and communication can be established immediately.

Type of transmission:

Cellular phones are designed to serve real-time full-duplex voice communications. Both the receiver and the transmitter are placed inside the device to provide two-way voice communication.

Speed and bandwidth:

Typical cellular networks presently run at approximately 20 Kbps, which is about half the speed of the standard dial-up modem.

Service range:

The cellular network is much more flexible and extensible than the paging network since it uses the cellular switching mechanism. Hence the service range is no longer limited by a single radio transmission station. Technically speaking the communication coverage can be expanded to wherever a cellular transmission tower can be placed.

Data processing capability:

The bandwidth for a cellular network is suited for low quality voice transmission. With add-on services it can also send and receive text messages and display low quality images.

Development and expectation

The concept of cellular communication was first introduced in 1947. However, due to technological limitations, the idea could not be implemented at that time. More research and development was done so that in 1982 the Advanced Mobile Phone Service (AMPS) became the first commercial cellular phone operator.

Although the development of cellular communication has been taking place for over half a century, due to several limitations it only really took off from the late 1990's. If the explosive tendency continues, within the next few years the legacy analogue cellular networks will be completely replaced by the digital cellular networks, service range will continue to expand, more services will be integrated into the cellular network, and the user group will grow steadily. Based on the research report from the *Washington Post*, by 2005 the number of cellular phone users will exceed 1 billion and their market value will reach USD 32 billion.

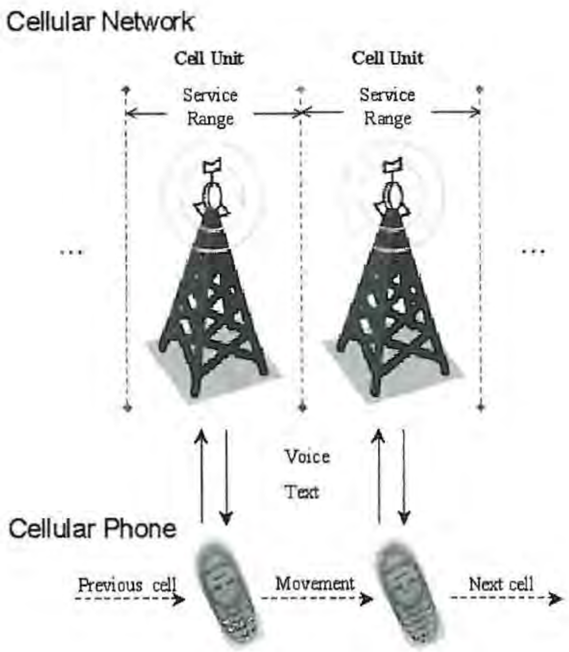


FIGURE 2.1.2-1 – CELLULAR NETWORK

Cellular Communication and mobile Web applications:

The nature of cellular communication is two-way and real-time, which is suitable for mobile Web applications. The wide adoption and popularity of cell phones makes cellular communication an effective, economical medium for implementation of mobile Web applications. In fact, nowadays most mobile Web applications are designed specifically for cell phones over cellular networks.

2.1.3 Mobile data communication

Besides all the other improvements such as network bandwidth and device processing power, what distinguishes mobile data communication from other communication technologies are the additional services, such as text messaging, voice communication, mobile video conferencing, high-speed wireless Internet access, remote data management, device interconnection, smart homes and appliances, and so on.

Type of transmission:

Mobile data communication supports multi-directional data transmission. There is a move from asynchronous messaging to real-time synchronized communication, from one to one voice communication to one to many video conferencing.

Speed and bandwidth:

Although transmission speed and bandwidth of mobile data networks vary from one implementation to another (for example, a technology known as Richochet provides a data rate of 128Kbps, the third generation cell phone is capable of transferring data at 2Mbps and some Wireless LANs can even handle a speed of 11Mbps or higher) they all promise a very high transmission speed, as compared to the traditional wireless technologies.

Service range:

Mobile data communication networks are built-up of a wide range of different multidimensional communication technologies, from cellular networks to satellites-based networks, from interconnected wireless local area networks (WLAN) to wireless wide area networks (WWAN), all of which complement one another. Loosely speaking the service range for mobile data communication is infinite.

Data processing capability:

With significantly increased processing power, local storages (ROM, RAM), and bandwidth; considerably improved display devices (bigger screens, higher resolution, colour support, etc.) and carefully designed user interfaces (touch screens, voice input, and so on, no longer only numeric keys) mobile data devices have the capability of processing a full range of data, from high quality voice streams to real time video streams, from text-based messaging to Web-based applications.

Development and expectation:

Since 1990, but especially over the last four or five years, mobile data communication technologies have been undergoing an explosive development. While initial development focused on digitising voice communication, current interest is in providing systems that support multi-data applications and add on services. "In 2001, only 3 percent of the 111 million mobile-phone subscribers in America used a mobile-data service, says Charles Golvin, senior analyst at Forrester Research. But by 2006, he predicts that two-thirds of the 180 million mobile-phone users will rely on data services. Most research suggests that greater acceptance of wireless data will occur as the phone networks roll out new data-based services, including access to online entertainment and m-commerce" [Nadel, Brian]. In the next few years mobile data communication will continue its high-speed innovation process, and will eventually be able to deliver a more reliable, simpler solution to business and consumer demands for faster, easier, and less expensive wireless data access and applications. A few years ago, during the era of the Internet revolution, there was a fresh and attractive concept called "taking the office home" or "work-from-home". Today a new revolution is about to begin, with the support of mobile data communication technologies. We can now extend this concept to, "taking the office everywhere" and "work-from-anywhere". It not only means that people will have a more relaxed and efficient working environment, but also implies that people will be able to access the information they are interested in, no matter where they are and what they are doing. In addition they can be notified instantaneously if there is a situation that needs attention. In this way corresponding responses and decisions can be made immediately. In the near future, the mobile data communication technologies will not only change the way people work, they will also change the way people live.

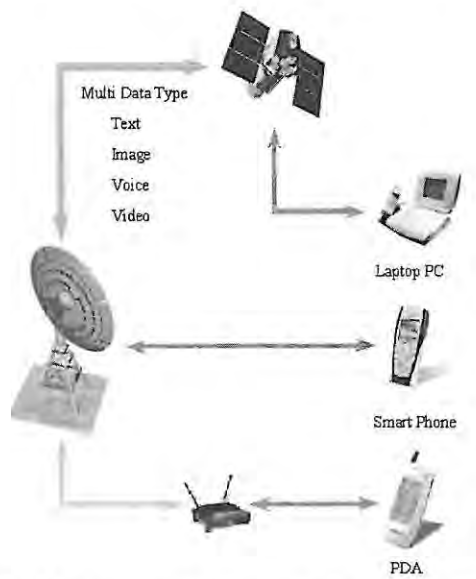


FIGURE 2.1.3-1 – MOBILE DATA COMMUNICATION NETWORK

Mobile Data Communication and mobile Web applications:

Increased processing power, transmission speed, multimedia support and so forth, make mobile data communication a platform capable of exerting all functionalities and potential of the next generation mobile Web applications. Mobile Web applications that are designed for and running on mobile data communication devices are no longer applications with a few lines of text and some unclear monochromic pictures. At a high level, mobile devices can be information consumers, data processors and providers at the same time.

2.1.4 Wireless communication comparison and summary

After having introduced these wireless communication technologies respectively, we can put them together and compare them.

A general comparison of wireless communications			
	Paging	**Cellular**	**Broadband**
Data type	Text	Voice, Text	All data ranges (video, data, etc.)
Speed	10Kbps (max)	20+Kbps (average)	128+Kbps (min)
Service range	Radio coverage (Limited mobility)	Cell switch (High mobility)	Infinite (Full mobility)
Transmission	One way	Two-way	Multi-way

TABLE 2.1.4-1 – A GENERAL COMPARISON OF WIRELESS COMMUNICATIONS

All these wireless communication technologies coexist today and complement one another in terms of functionality and applying scope. They target different markets and different user groups. At the present time the paging communication is still undergoing improvements. However, the market is shrinking since paging communication is becoming a part of services offered by other wireless communication technologies and it is likely that it will fade away from the market eventually. Cellular communication is the dominating wireless communication technology. The cellular networks keep on expanding with the number of users continuously growing, a tendency that is likely to continue over the next few years. Broadband mobile data communication is still a very new technology. It defines and presents the future of wireless communication – an integrated high speed wireless network with ubiquitous connectivity, which provides a full range of wireless services for multi-type of data including voice, video, text, application and so forth. In the future, with the support of broadband mobile data communication, voice and text will not be the only forms of service over the wireless network. And the wireless network will become a platform for business, management, entertainment and much more.

2.2 Mobile web protocols

There are quite a few mobile Web solutions in existence today, with some others currently being developed to fulfil the increasing demand for more efficient information exchange and retrieval, and customer satisfaction. In the previous section we introduced a number of wireless communication technologies, which provide the infrastructure and the application platform for wireless Internet access and mobile Web applications. In the following section we will have a look at the influential protocols that are widely adopted by the mobile devices manufacturers and large customer groups. They are: Wireless Access Protocol (WAP) developed by WAP Forum, NTT DoCoMo's i-mode, Sun Microsystems' Java mobile solution – Java2 Mobile Edition (J2ME), and .NET Mobile solution from Microsoft. In this chapter we will only discuss these protocol architectures at a high-level. The SUN Java Mobile Solution and the Microsoft Mobile Solution will be emphasized, discussed and compared in detail through all subsequent chapters.

2.2.1 Wireless access protocol (WAP)

2.2.1.1 WAP protocol architecture

WAP, namely the Wireless Access Protocol, was first published in April 1998 by WAP Forum[2]. It provides connectivity and accessibility to the Internet for mobile devices. WAP is designed to accommodate a wide array of wireless networks such as Cellular Digital Packet Data network (CDPD), Time Division Multiple Access network (TDMA), Code Division Multiple Access network (CDMA), Global System for Mobile Communication network (GSM), and so on. WAP is built-up of several underlying protocols, namely Wireless Application Environment (WAE), Wireless

[2] WAP Forum is an industry association of wireless device manufacturers, service providers and software companies, founded in July of 1997 by Ericsson, Motorola, Nokia and Phone.com.

Session Protocol (WSP), Wireless Transaction Protocol (WTP), Wireless Transport Layer Security (WTLS), Wireless Datagram Protocol (WDP) and Wireless Markup Language (WML) [See figure 2.2.1-1].

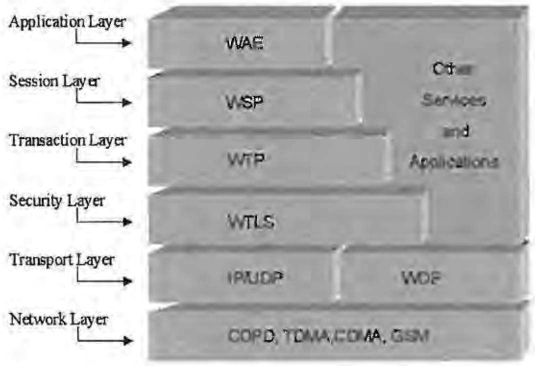


FIGURE 2.2.1-1 – WAP PROTOCOL LAYOUT

2.2.1.2 WAP network communication architecture

Nowadays the majority of Web-content is developed and deployed under an HTML standard and transferred using HTTP. Even though HTML works fine with personal computers, it is not suitable for mobile devices, which commonly have very limited display capabilities. By using WAP the original Web content can be transformed into a suitable format and thereby displayed on client devices. For example: A user makes an access request to a Web site using his WAP-enabled cell phone. The request goes through a WAP gateway and is redirected to the Web site. The Web site receives the request and sends back the relevant content in HTML format. The WAP gateway then receives the content and translates it into WML, and sends it to the user. Security Sockets Layer (SSL) encryption is used for the connection from WAP gateway to the Web site, and Wireless Transport Layer Security protocol (WTLS) encryption is used for the wireless transmission of content radio packets between the user and the WAP gateway to ensure the wireless security [see figure 2.2.1-2].

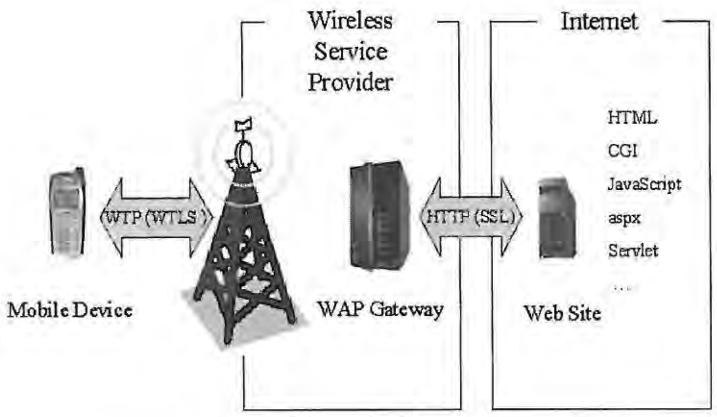


FIGURE 2.2.1-2 – HIGH LEVEL WAP ARCHITECTURE

2.2.1.3 Language for WAP mobile Web application - WML

WAP uses Wireless Markup Language (WML) to represent Web content. WML is defined by the WAP Forum and is specifically designed for the access of the Internet for wireless mobile devices, such as cellular phones. WML uses a card-like structure to organize the content of on-screen data representation [See figure 2.2.1-3]. It is written in XML[3]. [Please refer to code 2.2.1-1 for a WML page sample].

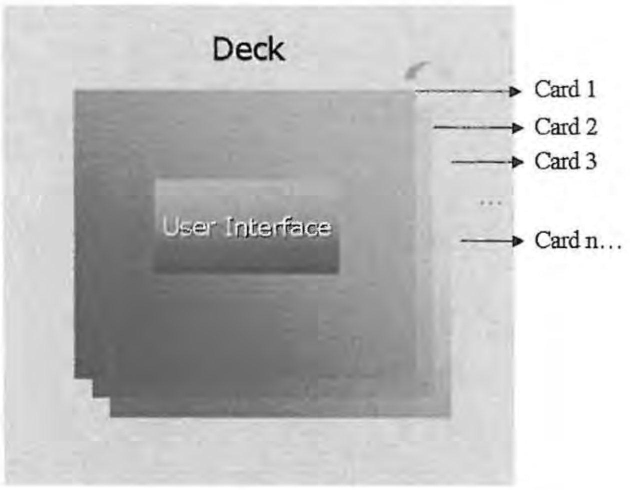


FIGURE 2.2.1-3 – WML CARD STRUCTURE

Sample Code:

```
<? xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml-1.1.xml">

<wml>
     <card id="card1" title="A simple WML page">
          <p>
               Hello WML!
          </p>
     </card >
</wml>
```

CODE 2.2.1-1 – WML CODE SAMPLE

[3] XML stands for eXtensible Markup Language. It is a World Wide Web Consortium standard for information exchange and data storage.

2.2.2 imode

2.2.2.1 imode introduction

Imode is a wireless data service solution developed by NTT DoCoMo[4] - a Japanese company. imode provides a wide range of Internet-based services, such as Web site browsing, email, music, news and so on. Although imode services are only deployed in Japan, it is by far the most successful mobile Internet service model as compared to other competing technologies. With its rational fee charge model, imode is the cheapest wireless data service in the world, with monthly subscription fees at around USD3.00, and additional data services being charged based on the data traffic, which is around USD0.003 per 128-bit data package. It has the largest number of subscribers in the world– more than twenty million in 2002. There are more than 40,000 imode compatible Web sites in existence all over Japan. Driven by the large and profitable market, a number of companies, such as Ericsson, Nokia, Oracle, AOL and so on, have anticipated the development of imode services and recently imode has started expanding into the European market.

2.2.2.2 imode network communication architecture

Imode runs on wireless packet switching networks with a data transfer speed of 9.6 kbps. All imode data traffic is under the monitoring of DoCoMo centres, [See figure 2.2.2-1]. DoCoMo servers and gateways listen for all requests and responses on all imode channels. Thus, DoCoMo centres provide unified high-level controls over imode users and services provided, such as redirection requests, authentication, authorization, access control, data encryption, fees charge, etc. For instance, if a user makes an access request to an imode compatible Web site, which is registered with DoCoMo centre, using his imode mobile device, the request first goes through a DoCoMo gateway and is then redirected to the Web site the user requested. The Web site receives the request and sends back the contents, which the DoCoMo gateway receives and then sends to the user. At the same time the amount of data packets transferred is recorded, so the user is charged accordingly.

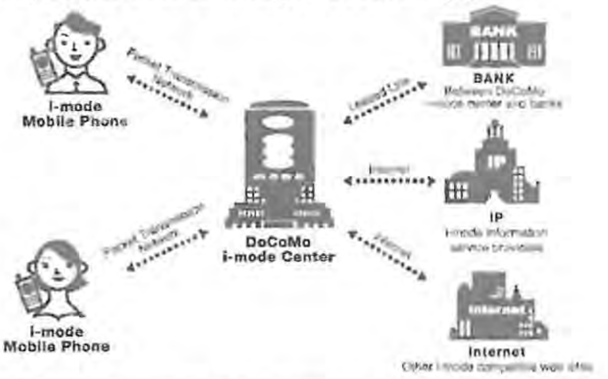


FIGURE 2.2.2-1 – HIGH-LEVEL IMODE NETWORK DIAGRAM [NTT DOCOMO]

[4] The "i" in imode means "information" and the "DoCoMo" in Japanese means "everywhere".

2.2.2.3 Language for imode mobile Web application – cHTML

Imode uses compact Hyper Text Markup Language (cHTML) to deliver the Web content. cHTML is a subset of HTML, a reduced version of HTML [Please refer to Code 2.2.2-1 for a cHTML page sample]. Hence, it supports a number of HTML tags and several binary formats, GIF for instance, and can also be displayed properly on any standard HTML Web browser without any additional modification.

Sample Code:

```
<HTML>
    <HEAD>
        <TITLE>
            A simple cHTML Page
        </TITLE>
    </HEAD>
    <BODY>
        Hello cHTML!
    </BODY>
</HTML>
```

CODE 2.2.2-1 – cHTML CODE SAMPLE

2.2.3 J2ME

2.2.3.1 Java technology:

Java technology has come a long way. It was originally designed for the development of embedded systems in set top boxes for home appliances. Before long Java moved into personal computers and was then widely used as a robust server-side technology. During the last few years Java technology has moved back into devices, but this time instead of home appliances the targeted systems are mobile devices – cell phones, PDAs etc. It has taken off along with the Internet revolution, so that today's Java technology consists of a full range of different but closely related technologies, which are clearly divided by their functionalities and applying scope. These technologies are Java 2 Platform Enterprise Edition (J2EE), Java 2 Platform Standard Edition (J2SE), Java 2 Platform Mobile Edition (J2ME), Java Card and so on. [See figure 2.2.3-1].

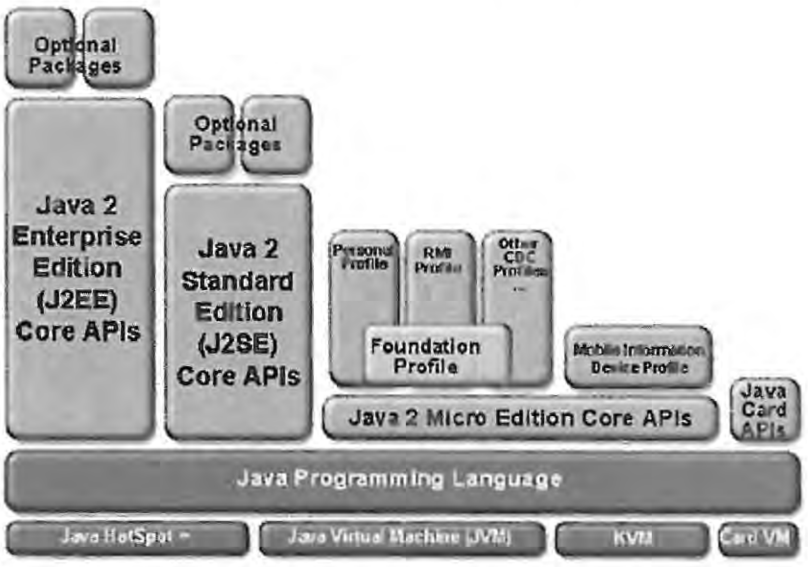

Optional Packages
Optional Packages
Java 2 Enterprise Edition (J2EE) Core APIs
Java 2 Standard Edition (J2SE) Core APIs
Personal Profile
RMI Profile
Other CDC Profiles ...
Foundation Profile
Mobile Information Device Profile
Java 2 Micro Edition Core APIs
Java Card APIs
Java Programming Language
Java HotSpot ™
Java Virtual Machine (JVM)
KVM
Card VM

FIGURE 2.2.3-1 – J2ME AND JAVA TECHNOLOGY FAMILY [SUN MICROSYSTEMS]

Java is highly portable, which gives programmers a standard way to write code that can be executed over a wide range of devices despite the underlying operating system. At the same time Java provides a mature, robust and secure sustenance for network programming. With its natural benefits, Java is an idea choice for developing wireless applications for various mobile devices. This is where the Java 2 Platform Mobile Edition (J2ME) comes into play.

2.2.3.2 J2ME:

Basically, J2ME is a subset of the Java 2 Platform Standard Edition (J2SE), which is itself a subset of the Java 2 Platform Enterprise Edition (J2EE). In fact J2ME is neither a specific piece of software nor has a specific specification. It can be considered as a collection of software packages that contains a set of Java application programming interfaces (API) and the J2ME runtime environment. It provides a modular, scalable architecture that is designed specifically for mobile devices.

Now let us have a look at the official definition. "The Java 2 Platform, Micro Edition is the edition of the Java 2 platform targeted at consumer electronics and embedded devices. The J2ME technology consists of a virtual machine and a set of APIs suitable for providing tailored runtime environments for consumer and embedded electronics. The J2ME technology has two primary kinds of components--configurations and profiles. The J2ME technology has two design centres--things that you hold in your hand and things you plug into a wall. These design centres have different qualities that are optimised for in the virtual machine and low-level libraries themselves. Configurations are composed of the two low-level APIs and optimised virtual machines targeted at two broad categories of devices: (1) those with 128-512 K of memory available for the Java technology environment (and applications) and (2) those with 512 K + available for the Java technology environment (and applications), [See figure 2.2.3-2]. Configurations are nestable, so that any software able to execute on a less capable configuration is able to execute on a more capable one. A profile is a specification that details the Java technology APIs, built on top of and utilizing the underlying Configuration, necessary to provide a complete runtime environment for a specific kind of device. Profiles specify both APIs and the Configuration (if offered). A profile must be complete in the sense that an application written to the specification can execute in the specified Java technology environment without the addition of other Java classes (as opposed to Optional Packages) [Please refer to figure 2.2.3-2]. Profiles can be thought of as selecting classes from APIs to form a complete environment. Profiles are designed and integrated to meet the needs of specific industry segments" [Sun Microsystems].

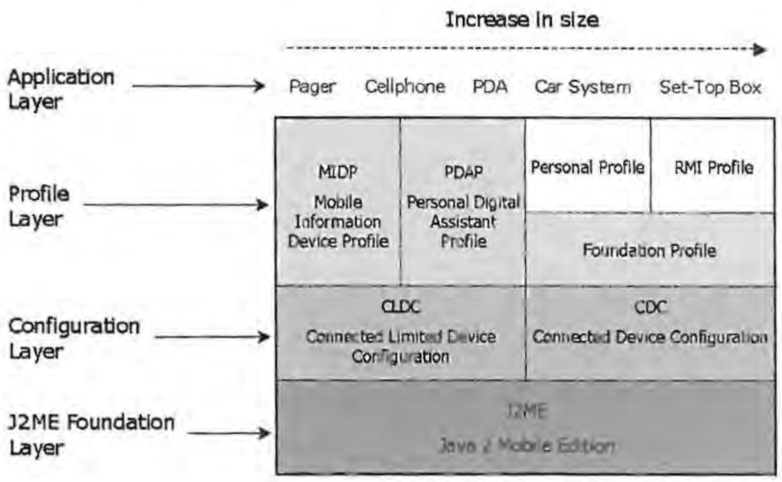

Increase in size
Application Layer
Pager
Cellphone
PDA
Car System
Set-Top Box
Profile Layer
MIDP Mobile Information Device Profile
PDAP Personal Digital Assistant Profile
Personal Profile
RMI Profile
Foundation Profile
Configuration Layer
CLDC Connected Limited Device Configuration
CDC Connected Device Configuration
J2ME Foundation Layer
J2ME Java 2 Mobile Edition

FIGURE 2.2.3-2 – J2ME ARCHITECTURE: CONFIGURATIONS AND PROFILES

2.2.3.3 J2ME mobile Web application:

Mobile Web solutions play a key role in J2ME technology. Via J2ME mobile Web solutions people can benefit from the rich content of the Internet.

At a high level the common J2ME Web systems use a three or n tiers architecture, [Please see Figure 2.2.3-3], which includes the front tier – client mobile device, the middle tier – Web server or WAP gateway, and the backend tier – the static database or other content. For example, a user uses his J2ME enabled mobile device, say a Java phone, to request a certain Web content. The request goes through the middle tier gateway and is directed to the content provider, which can be a static database or an HTML page. It can also be some dynamic content generated by server-side technologies such as Java Servlets or Enterprise JavaBeans (EJBs). The content provider then sends response content back to the gateway. At this stage the response is in the original format, which can be HTML or XML, and because most mobile devices do not support these formats the content must be "translated" before it goes back to the client device. The middle tier server or gateway receives and "translates" the content based on the requesting mobile device capability and then sends the "translated" content (which can be WML, cHTML, or any other format) to the client. Finally, the content is displayed by an embedded micro browser or a J2ME application on the mobile device.

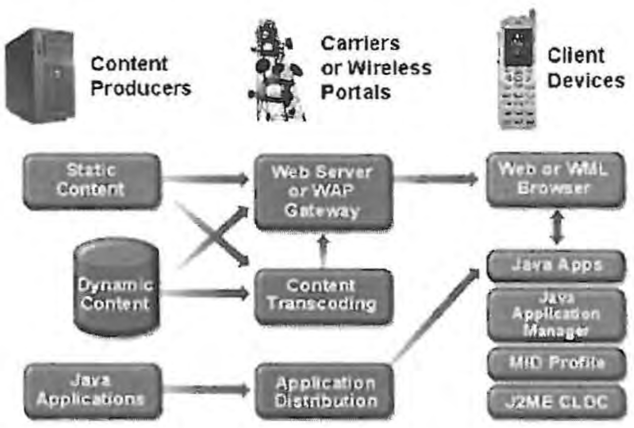


FIGURE 2.2.3-3 – HIGH-LEVEL J2ME MOBILE WEB APPLICATION ARCHITECTURE [Sun Microsystems]

2.2.3.4 Language for J2ME mobile Web application – Java

J2ME mobile Web applications use Java and a set of APIs provided by the MIDP to construct mobile applications [Please see Code 2.2.3].

Sample Code:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class J2MEMobile extends MIDlet {
        private Form myForm;
        private Ticker ticker;

        public void startApp(){
                myForm = new Form();
                ticker = new Ticker("Hello J2ME!");
                splashForm.setTicker(ticker);
                Display.getDisplay(this).setCurrent(myForm);
        }
        public void pauseApp() {}
        public void destroyApp(boolean unconditional) {}
}
```

CODE 2.2.3-1 – JAVA CODE SAMPLE

2.2.3.5 J2ME development and future expectation:

J2ME receives much attention from various industries, and most major mobile device manufacturer and carriers, such as Motorola. Ericsson, Nokia, Siemens, Sharp, Philips Electronics, NEC, Fujitsu, Hitachi, Mitsubishi Electric, Samsung Electronics, LG, France Telecom, NTT DoCoMo, Orange and Vodafone are involved in the development of J2ME. With such a strong support from industry, a great number of Java-enabled mobile devices find their way into the market [Please refer to Appendix B for more information], and more java mobile applications are currently being written.

The latest version of the standard set of technologies for Java-capable mobile devices – the MIDP2.0 - has recently been finalized and was announced by Sun Microsystems at Telecom Asia 2002. J2ME is a very new technology, and is still rapidly developing. A lot of specifications still remain to be written. Sun and its partners are working together to expand the potential of J2ME through the Java Community Process (JCP)[5].

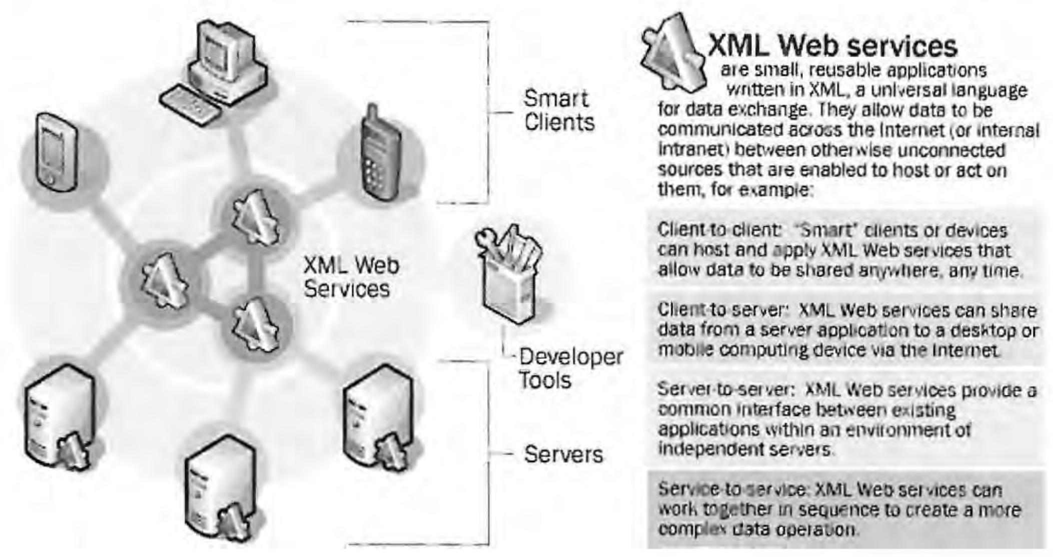
2.2.4 Microsoft .NET mobile solution

2.2.4.1 Microsoft .NET technology:

"Microsoft .NET is a set of Microsoft software technologies for connecting your world of information, people, systems, and devices. It enables an unprecedented level of software integration through the use of XML Web services: small, discrete, building-block applications that connect to each other—as well as to other, larger applications—via the Internet." [Microsoft]

.NET is not a simple extension of the PC based computing architecture; it offers the new concept of extending distributed computing throughout the Internet. If it lives up to its promise, .NET will provide ubiquitous Internet-oriented applications and services, such as Web services and services for mobile/wireless devices, with security and privacy at the core. These services are intended to stretch across devices, languages, and operating systems. At the same time .NET is a development environment that provides tools, which allow developers to build applications and services in a rapid prototyping fashion [Please see figure 2.2.4-1].

[5] JCP is the way the Java platform evolved. It is an open organization of international Java developers and licensees whose charter is to develop and revise Java technology specifications, reference implementations, and technology compatibility kits.



XML Web services are small, reusable applications written in XML, a universal language for data exchange. They allow data to be communicated across the Internet (or internal intranet) between otherwise unconnected sources that are enabled to host or act on them, for example:

Client-to-client: "Smart" clients or devices can host and apply XML Web services that allow data to be shared anywhere, any time.

Client-to-server: XML Web services can share data from a server application to a desktop or mobile computing device via the Internet.

Server-to-server: XML Web services provide a common interface between existing applications within an environment of independent servers.

Service-to-service: XML Web services can work together in sequence to create a more complex data operation.

FIGURE 2.2.4-1 – CONCEPTUAL .NET ARCHITECTURE [MICROSOFT]

In general terms .NET is a long-term strategic concept, a brand name from Microsoft, and a taste of the future of the Internet. More technically, .NET is made up of XML, the Web Service, and the .NET framework.

1) XML (eXtensible Markup Language) has become a standard for data exchange and storage.

2) Web Service is a concept that facilitates interactivity between different devices, such as PCs, PDAs, and cell phones, across the Internet. SOAP (Simple Object Accessing Protocol) is used as the communication standard while UDDI (Universal Discovery Description and Integration) is used to register and find Web services.

3) The .NET Framework is a new generation development and deployment platform. It provides a unified Base Class Library, and a CLR (Common Language Runtime) environment that enables Web services to be developed in different programming languages.

Functionalities of .NET	
1	Provides a .NET architecture, built on XML, SOAP, and UDDI.
2	Provides interconnection between Websites and contents.
3	Provides a wide range of .NET services.
4	Provides .NET Interfaces for developers.
5	Provides inter-operability on a range of mobile devices.

TABLE 2.2.4-1 – .NET FUNCTIONALITIES

2.2.4.2 .NET mobile:

The .NET mobile layer in the previous table is critical to the entire strategic concept of the .NET, which promises to provide services to any authorized person at any time, in any place, and on any device. The .NET mobile solutions are built upon these layers, and some newly emerging mobile devices have already started supporting these standards.

.NET Mobile is an extension of Microsoft ASP.NET and Microsoft's .NET Framework. .NET Mobile Web Applications can be transferred on any type of wireless network and can be accessed just like any normal Web application. The Web application server, however, that serves the application must be .NET compatible. That is to say the server must have .NET Framework installed. When a request to the .NET Mobile Web Page is received by a Web application server, the server transforms the page into a different output format, such as WML or cHTML, based on the requesting mobile device's capability [Please see figure 2.2.4-2].

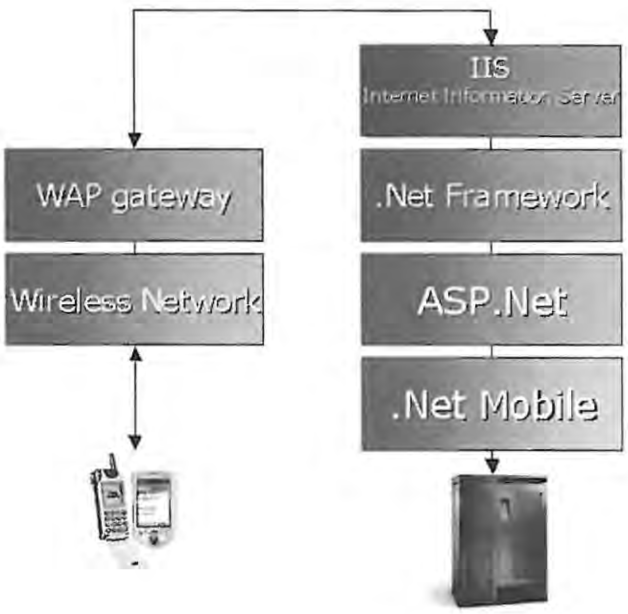


FIGURE 2.2.4-2 – .NET MOBILE WEB NETWORK ARCHITECTURE

2.2.4.3 Language for .NET mobile Web application –ASP.NET mobile

.NET mobile Web application contains a series of server-side Web form controls to build mobile Web applications for wireless mobile devices. Web pages composed with these Web form controls are written in XML syntax [Please see code 2.2.4-1].

Sample code:

```
<%@ Page language="c#" Codebehind="default.aspx.cs"
Inherits="System.Web.Mobile" AutoEventWireup="false" %>

<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile%>

<body>
<mobile:form id="Form1" runat="server">
<mobile:Label id="Label23" runat="server" >
                    Hello .Net Mobile
               </mobile:Label>
</mobile:form>
</body>
```

CODE 2.2.4-1 –MOBILE ASP.NET CODE SAMPLE

2.2.5 Mobile Web solutions comparison and summary

All the mobile Web solutions introduced currently use a wide range of different technologies, since they have various architectures, run on different models and transfer data on different networks. The resulting overall picture looks as follows.

A general comparison of Wireless Web Application Protocols				
	WAP	**imode**	**J2ME**	**.NET**
Network	Any	CDPC	Any	Any
Speed	Any	9.6Kbps	Any	Any
Language	WML	CHTML	Java	ASP.net
Device	WML enabled	cHTML compatible	JVM installed	Any devices
Market Adoption	US and Euro	Japan	World wide	Just started

TABLE 2.2.5-1 – WIRELESS WEB APPLICATION PROTOCOLS

In the ever-crowded wireless Internet access marketplace, it is unlikely that any single mobile Web technology will dominate the market. There are, however, numerous facts that are clearly affecting the adoption and success of a mobile Web technology. They include technical elements, such as media capability and service cover range, and non-technical elements, such as business models, consumer expectation and so on. The most advanced technology is not always the most successful business-wise. The one factor that is essential to the success of any mobile Web provider is what kind of application or service can be provided and how well and efficiently it can be provided.

2.3 Chapter review

This chapter is an overview of the existing wireless communication technologies and mobile Web protocols. In the first part of this chapter we introduced three major wireless communication modes currently in existence, namely, paging communication, cellular communication and mobile data communication. The introduction covered the network architecture, history and development expectation of each of the technologies. In the second part of the chapter we took a look at four wireless protocols that are used to create mobile Web applications and services, which include WAP, imode, J2ME and .NET Mobile. In the next chapter we will go deeper into the creation of mobile Web applications in development environments using J2ME and .NET Mobile solution.

Chapter 3: Developing J2ME Mobile Web Applications

- *"The goal of Computer Science is to build something that will last at least until we've finished building it"* [GATs' Guide].

Two mobile Web applications have been developed for this project. One was developed using J2ME, which is presented in this chapter, the other used .NET, which will be discussed in next chapter. Therefore a true working development and deployment environment is provided for objective comparisons of the two mobile Web solutions.

In this chapter a J2ME mobile Web application – *"Rhodes University Mobile Meal Booking System"* – is introduced and analysed. This application allows students to manage their meals using their cell phones and to do things like book/un-book meals, change diets, view menus, send feedback emails, and so forth [Source code, related material and video clip demos can be found on the accompanying CD-ROM].

3.1 Development platform and environment – Hardware

Both of the applications share the same hardware platform and the network environment, as presented in figure 3.1.1-1.

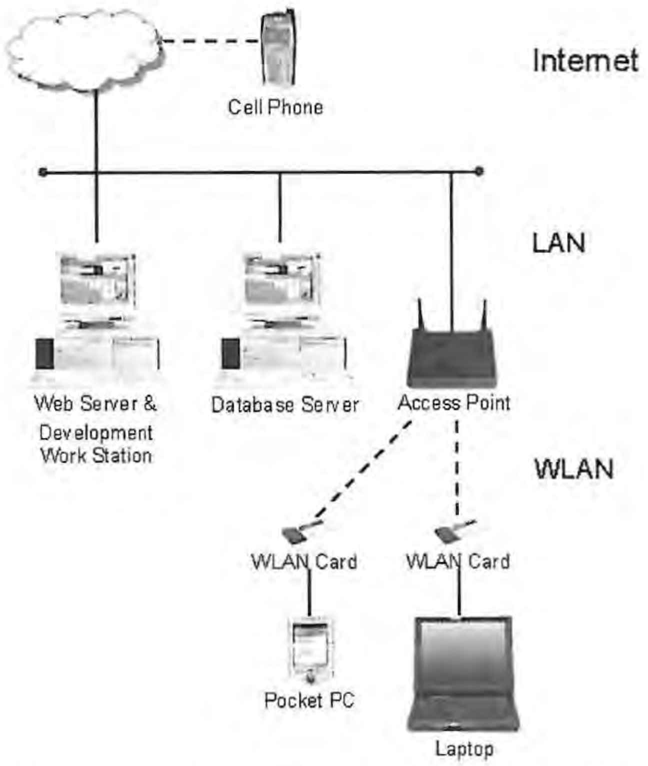


FIGURE 3.1.1-1 – HARDWARE AND NETWORK ENVIRONMENT

Servers and workstations:

Web server: Pentium III 450 dual processor, 512 Mbytes RAM.
Database server: Pentium III 700 processor, 128 Mbytes RAM.
Development workstation: same as the Web server.

Client devices:

Laptop: Compaq Armada 1750 Pentium II 333MH processor, 128 Mbytes RAM.
PDA: Compaq Pocket PC Model 3630, ARM SA1110, 32 Mbytes flash memory.
Cell Phone: Motorola i85s emulator and Openwave 4.0 emulator.

Network connectivity:

LAN 100Mbps local area network connection.
WLAN card: Compaq WL110, 11Mbps wireless connection.

3.2 The *"Mobile Meal Booking System"*

"*Rhodes University Mobile Meal Booking System*" is a J2ME mobile Web application. It allows users to manage their meal bookings through mobile devices, such as cell phones. The following sections conduct a detailed observation and analysis of the "*Mobile Meal Booking System*", which includes the development environment setting, the client-side development (front tier mobile component), the Web service development (middle tier Web component) and database design (backend tier data component).

3.2.1 Software installation and environment configuration

Setting up the software environment is the first and an important step of developing any application. It includes installation, configuration, investigation and optimisation. All software packages, including the Operating System used for development and deployment of the "*Rhodes University Mobile Meal Booking System*", are developed and licensed under the Open Source GNU General Public License (GPL) and/or Free Software model.

3.2.1.1 Installation & configuration

Operating System:
Mandrake[6] Linux 8.2, kernel version: 2.4.18-6mdksmp.

[6] Mandrake Linux is a GNU/Linux distribution supported by MandrakeSoft S.A. MandrakeSoft was born into the Internet in 1998 with its main goal being to provide an easy-to-use and friendly GNU/Linux system. The two pillars of MandrakeSoft are open-source and collaborative work. [Mandrake]

Application Development and Runtime Environment:

1) J2SDK and J2RE Version 1.4 (Disk space: 70 Mbytes)
2) J2ME_MIDPVersion 1.0 (Disk space: 24 Mbytes)
3) J2SDKEE Version 1.3.1 (Disk space: 25 Mbytes)

JRE is the runtime environment for all Java applications. J2SDK, J2ME_MIDP and J2SDKEE supply programmers with sets of development tools and bundles of precompiled classes (libraries) for the development of Java applications. The installation of J2ME_MIDP and J2SDKEE depends on JRE and J2SDK.

Installation:

To install these packages, uncompress the archive files, and copy them into any directory.

Configuration:

Edit $HOME/.bash_profile:

1) Add the "bin" and "lib" directory where those packages are installed.

```
PATH=./:/myShare/j2sdk1.4.0_02/bin:/myShare/j2sdk1.4.0_
02/lib:/myShare/j2sdkee1.3.1/bin:/myShare/j2sdkee1.3.1/lib:
/myShare/WTK104/bin:/myShare/WTK104/lib
```

CODE 3.2.1-2 –SETTING PATH VARIABLE

2) Create and export environment variables for each of the packages

```
JRE=/usr/java/j2re1.4.1_01
JAVA_HOME=/myShare/j2sdk1.4.0_02
J2EE_HOME=/myShare/j2sdkee1.3.1
J2ME_HOME=/myShare/WTK104

export JRE
export JAVA_HOME
export J2EE_HOME
export J2ME_HOME
```

CODE 3.2.1-2 –SETTING ENVIRONMENT VARIABLES

Web Server:

Apache Web Server Version: 2.0.42

The Apache Web server is the product of the "Apache HTTP Server Project"[7]. It is the most popular free Web server.

Installation:

It comes with the J2SDKEE.

Configuration:

No further configuration is required for this project.

Database Server:

1) MySQL Production Version: 3.23.52

"The MySQL database server is the world's most popular open source database. Its architecture makes it extremely fast and easy to customize. Extensive reuse of code within the software and a minimalistic approach to producing functionally-rich features has resulted in a database management system unmatched in speed, compactness, stability and ease of deployment. The unique separation of the core server from the storage engine makes it possible to run with strict transaction control or with ultra-fast transactionless disk access, whichever is most appropriate for the situation."[MySQL].

Installation:

It comes with the Mandrake Linux distribution.

Configuration:

Edit $HOME/.bash_profile and add the directory where MySQL server is installed into the PATH variable.

```
PATH=...:/usr/share/mysql:$PATH
```

2) JDBC-Mysql Driver 2.0.13 (Disk space: 1.5 Mbytes)

The JDBC Driver for MySQL was created by Mark Matthews[8]. It provides a set of Java Interfaces for programmers to handle the connectivity and operation of the MySQL database.

[7] The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards [Apache].

[8] Mark Matthews is one of the developers of the Open Source "MM.MySQL" driver project.

Installation:

Uncompress the archive file and copy the mysql-2.0.13-bin.jar file into $JAVA_HOME/jre/lib/ext directory, or any other directory (Configuration needed).

Configuration:

Edit $HOME/.bash_profile and add the path of mysql-2.0.13-bin.jar to an environment variable.

```
JDBC=/usr/jdbc

export JDBC
```

CODE 3.2.1-3 – SETTING JDBC ENVIRONMENT VARIABLE

IDE and Toolkits:

1) SunOne Studio Mobile Edition: (Disk space: 47 Mbytes)

SunOne Studio Mobile Edition is an IDE offered by SUN for free for programmers to develop Java mobile applications.

Installation:

SunOne Studio Mobile Edition provides a GUI installation interface.

Configuration:

Set up the Web server, proxy server and IDE layout options, and the location of J2ME Wireless Toolkit.

2) J2ME Wireless Toolkit (JWTK) Version 1.0.4 (Disk space: 24 Mbytes)

WTK is a free tool for compiling and testing the J2ME mobile applications. It also provides a series of mobile device emulators such as Motorola i85s and Java Palm.

Installation:

To install, just uncompress the archive and copy it into an arbitrary directory.
Configuration:

WTK uses the same configuration as the J2ME_MIDP.

3) J2EE Application Deploy Tool

J2EE Application Deploy Tool supplies a group of Graphical Interfaces to deploy J2EE applications.

Installation:

It comes with the J2SDKEE.

Configuration:

The J2EE Application Deploy Tool is configured automatically when running the J2EE server.

3.2.1.2 Investigation & optimisation

It takes up to 200 Mbytes disk space (Excluding the Operating System) to install all the software and the software is free of charge (Including the Operating System) and to build up the entire development platform.

In developing the client-sides of mobile Web applications the design of the Graphical User Interface (GUI) is essential. Unfortunately SunOne Studio Mobile Edition does not provide a GUI designer. The MySQL server offers only shell prompt interfaces for database operation and administration.

A workstation with 512 Mbytes RAM is sufficient for the program that has the highest requirements – the "SunOne Studio Mobile Edition" – to run smoothly. Most of the tools and the IDEs used for this project are written in Java. Hence, it can be transported directly to other Operating Systems such as SUN Solaris and Microsoft Windows without any modification. At the same time, however, because Java uses the Java Virtual Machine (JVM) to execute binary code instead of using Hardware to execute machine code, it is relatively slower to use a "pure Java" development environment. For example, it takes 35–40 seconds on average to launch the SunOne Studio IDE.

3.3 Developing the Client-side mobile component

3.3.1 MIDlet – the "Compact Applet" on mobile devices

The Java program that actually runs on J2ME enabled devices is called a MIDlet. The style and convention of a MIDlet is similar to that of a Java Applet or a Java Servlet. It is represented by inheriting and instantiating the javax.microedition.midlet.MIDlet class.

3.3.2 MIDlet life cycle

Each MIDlet has its own specific life cycle, as shown in figure 3.3.2-1, which is reflected in the methods and behaviour of the MIDlet class. The life cycle of any MIDlet is controlled and managed by special software – namely the application manager, which exists in every J2ME enabled device.

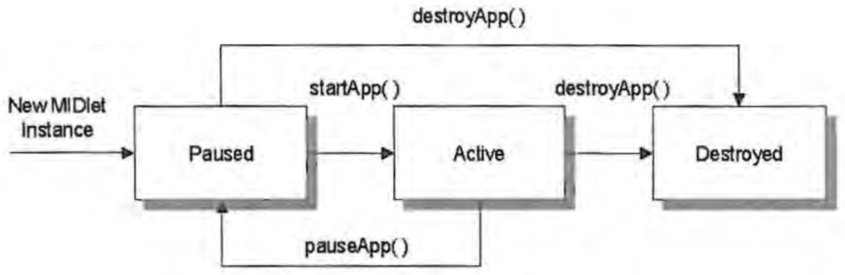


FIGURE 3.3.2-1 – THE GENERAL MIDLETS LIFE CYCLE

3.3.3 MIDlet packages and classes

3.3.3.1 CLDC /MIDP packages and classes

As shown in figure 3.3.3-1, standard CLDC/MIDP APIs contain seven packages:

1) User Interface Package: javax.microedition.lcdui;

2) Persistence Package: javax.microedition.rms;

3) Application Lifecycle Package: javax.microedition.midlet;

4) Networking Package: javax.microedition.io;

5) Core Packages: java.io, java.lang and java.util;

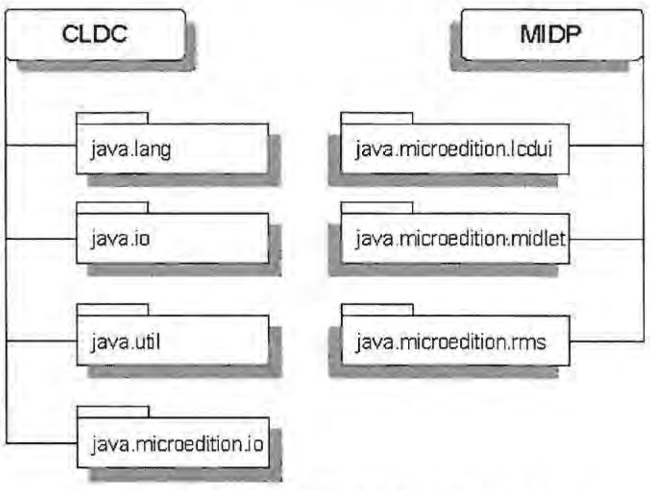


FIGURE 3.3.3-1 – THE CLDC/MIDP PACKAGES

The *"Mobile Meal Booking System"* is a mobile Web application. Therefore, this application focuses mainly on the usage of the javax.microedition.lcdui package and the javax.microedition.io package.

1) The javax.microedition.lcdui package provides a set of features for the implementation of user interfaces for Mobile Information Devices Profile (MIDP) applications.

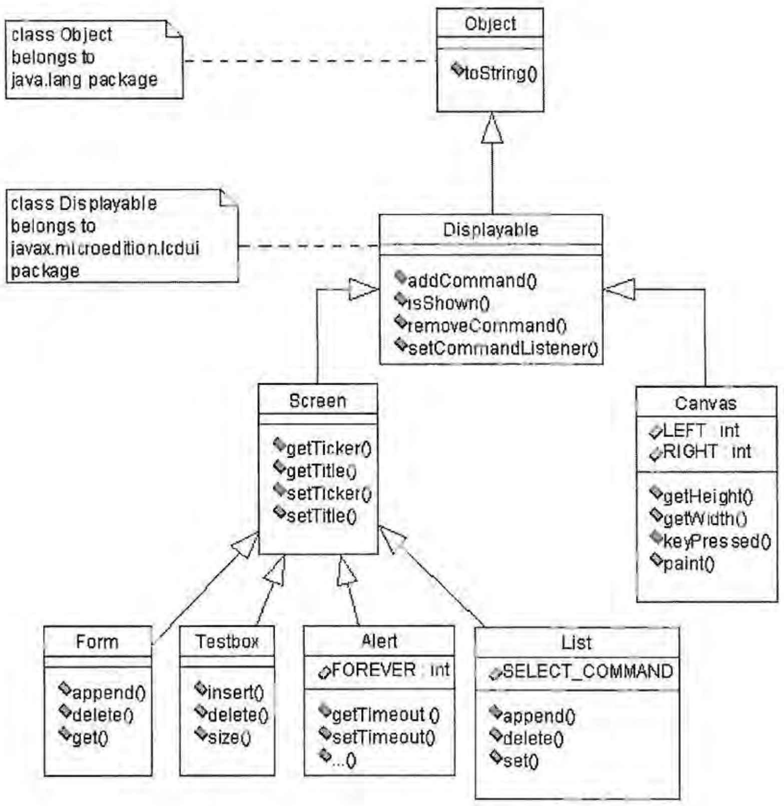


FIGURE 3.3.3-2 – J2ME-CLDC-MIDP GUI CONTAINER CLASSES

Form and List are two of the most commonly used top-level container[9] classes for the creation of user interfaces for the *"Mobile Meal Booking System"*. Other GUI controls such as Command, StringItem and Image are to be added onto the container. Code 3.3.3-1 demonstrates the usage of the List container.

Code:

```
private void bookingMenu(){

    //initial list and list items
    final String []  strList = {"1. Book meal", "2. Unbook meal", ...};

    final List menuList = new List("", List.IMPLICIT, strList, null);

    //add button onto the List
    menuList.addCommand(cmdOK);

    //declaring and binding the event handler to the list
    menuList.setCommandListener(new CommandListener() {
        public void commandAction(Command cmd, Displayable dis){
            ...
            }//commandAction()
        }//CommandListener
    );//addCommandListener()

    //set display to current list
    Display.getDisplay(this).setCurrent(menuList);
}//bookingMenu()
```

CODE 3.3.3-1 – TOP LEVEL GUI CONTAINER

2) The javax.microedition.io package includes networking support based on the GenericConnection framework from the Connected Limited Device Configuration (CLDC) [Please see figure 3.3.3-2].

[9] A container is a GUI component with the capability of containing other GUI components.

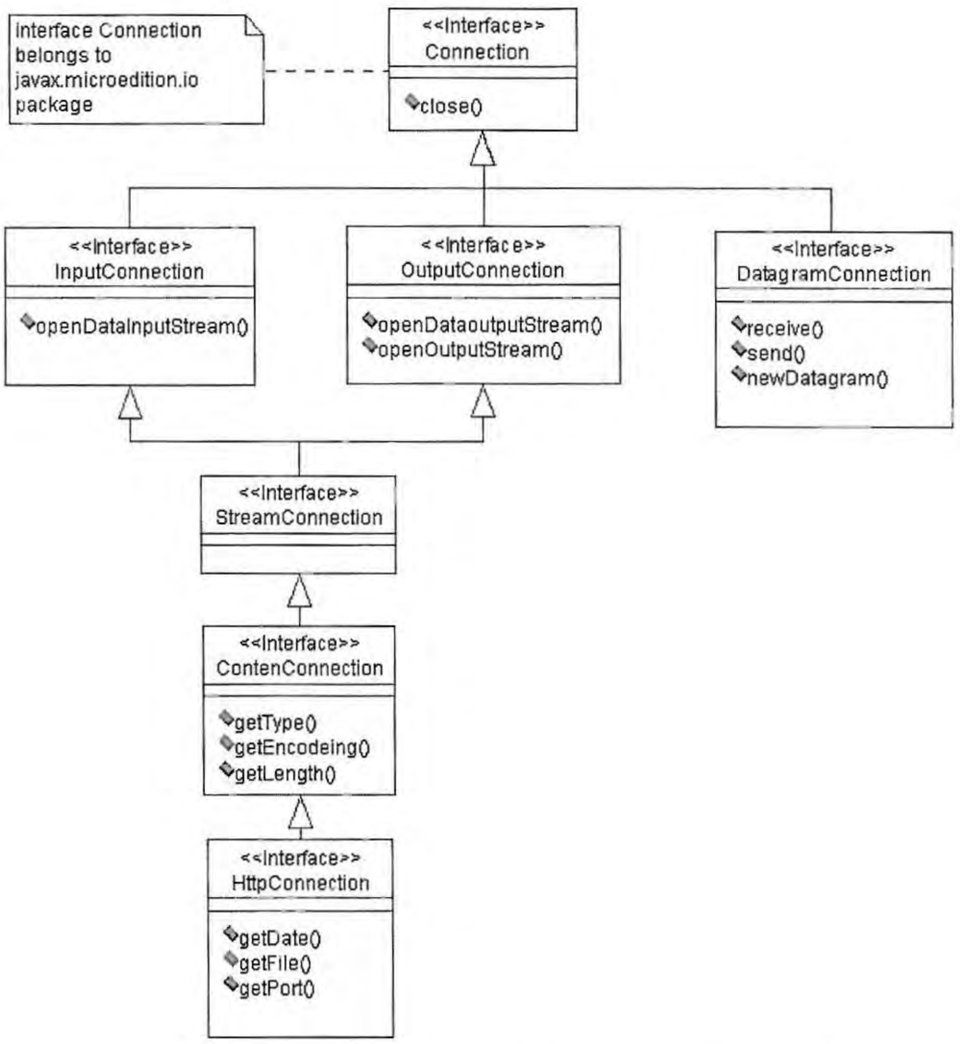

Interface Connection belongs to javax.microedition.io package
<<Interface>> Connection
close()
<<Interface>> InputConnection
openDataInputStream()
<<Interface>> OutputConnection
openDataoutputStream()
openOutputStream()
<<Interface>> DatagramConnection
receive()
send()
newDatagram()
<<Interface>> StreamConnection
<<Interface>> ContenConnection
getType()
getEncodeing()
getLength()
<<Interface>> HttpConnection
getDate()
getFile()
getPort()

FIGURE 3.3.3-3 – J2ME-CLDC-MIDP CONNECTION INTERFACES

Being a Web application, the *"Mobile Meal Booking System"* uses only the HttpConnection interface from the javax.microedition.io package to set up the HTTP connection from the MIDlet to the Web component [see code 3.3.3-2 for an example].

```
...
try{
      HttpConnection connection = (HttpConnection)Connector.open(URL);
      }
catch(IOException e){
      e.printStackTrace();
}
...
```

CODE 3.3.3-2 – SETUP HTTP CONNECTION

3.3.3.2 "Mobile Meal Booking System" classes hierarchy overview

Figure 3.3.3-4 demonstrates the hierarchy and relationships of the classes of the MealBookingMIDlet in the *"Mobile Meal Booking System"* under the Java Platform.

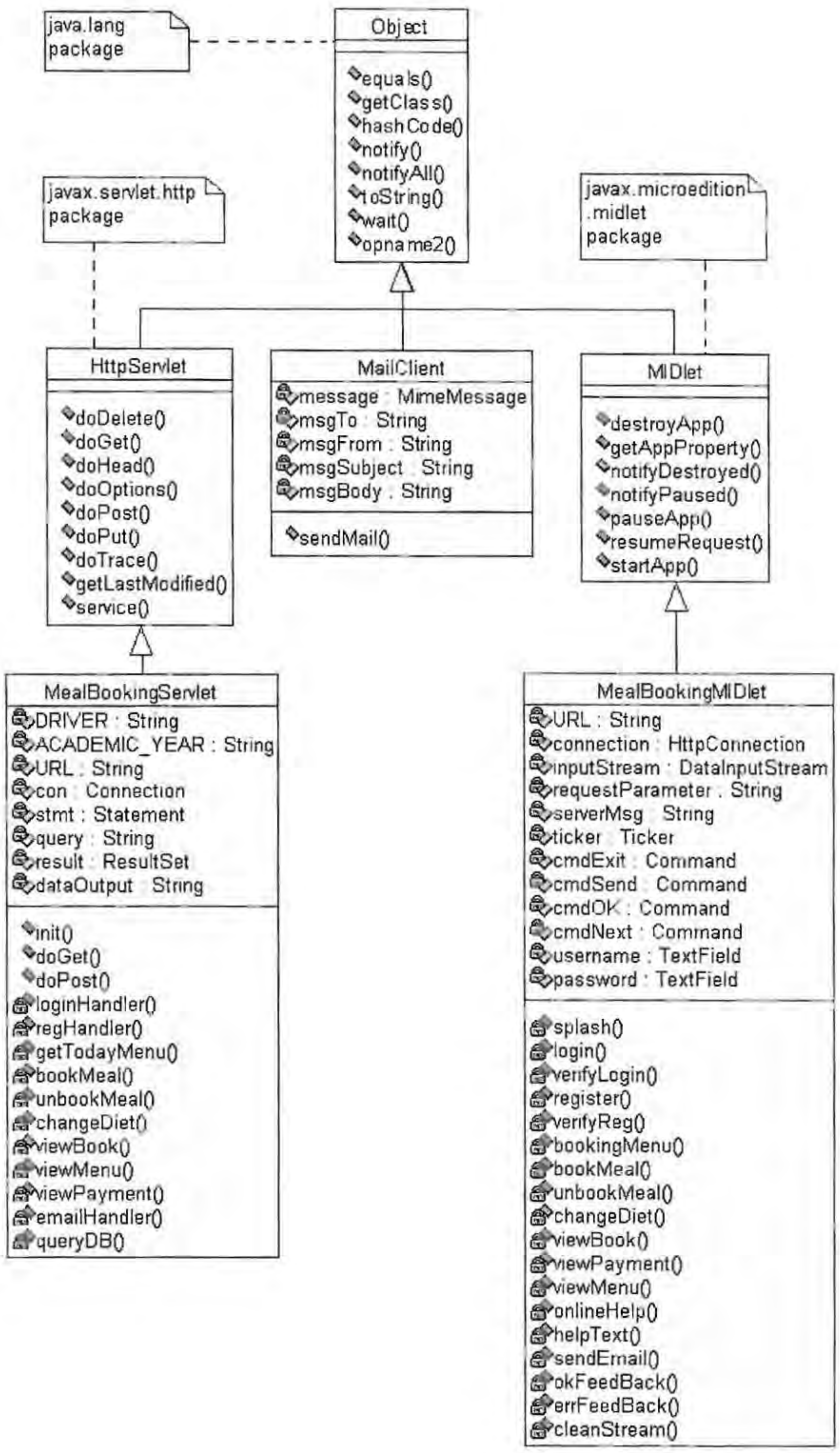


FIGURE 3.3.3-4 – *"MOBILE MEAL BOOKING SYSTEM"* CLASSES HIERARCHY

3.3.4 MIDlet development life cycle

Generally the development of a MIDlet includes four major stages: designing, programming, testing and deploying. Designing can be divided into four steps: defining design goals, categorizing functional modules, abstracting interactions between modules and designing user interfaces and GUI layout. The programming stage can be further classified into three steps: coding, compiling and preverifing [Please see figure 3.3.4-1]. The *"Mobile Meal Booking System"* can be classified in this manner.

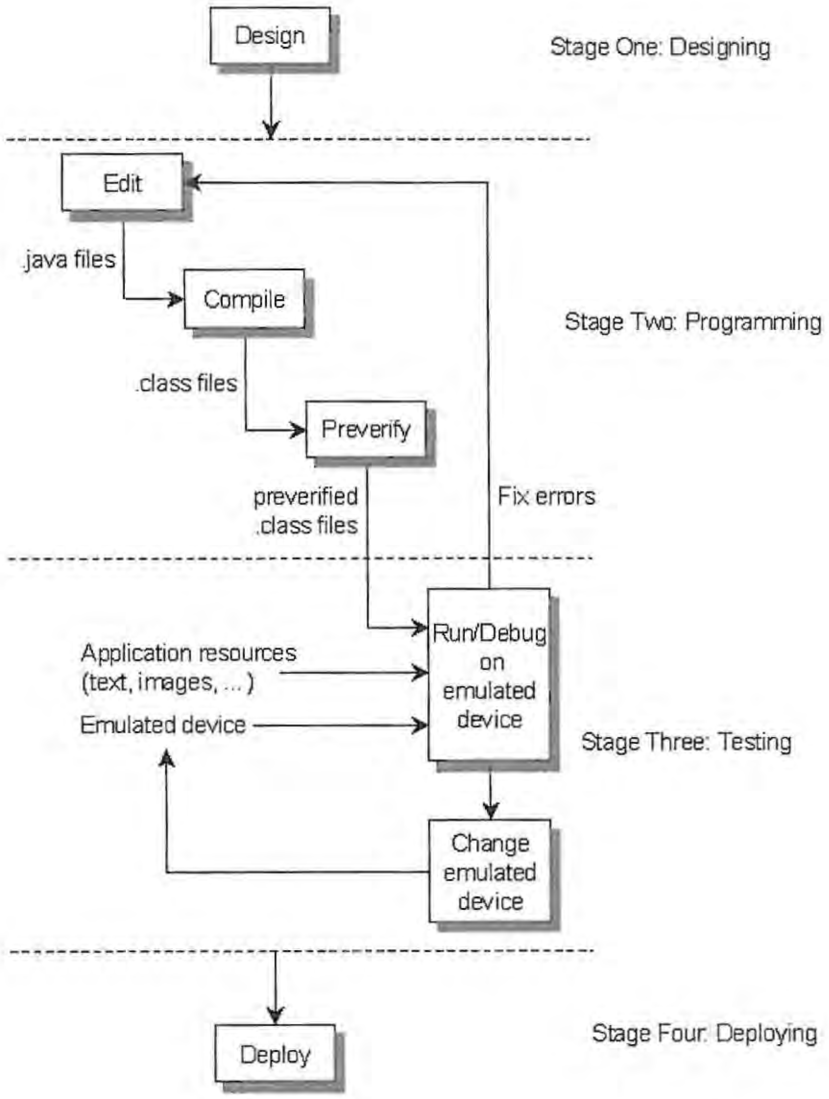


FIGURE 3.3.4-1 – THE GENERAL MIDLETS DEVELOPMENT LIFE CYCLE

3.3.5 Developing MIDlet for the "Mobile Meal Booking System"

3.3.5.1 STAGE ONE: Design the MealBookingMIDlet

Step 1: Define design goals:

To start we shall clarify the design, goals and expectations of a completed system.

Goals:

1) Develop a MIDlet application that fulfils all requirements and performs all functions as a mobile client for the *"Meal Booking System"*.

2) The MIDlet must be able to interact with a Web component through the HTTP connection.

3) The user interface must be straightforward and easy to understand.

4) The GUI interfaces and layout must suit mobile devices.

Step 2: Categorize functional modules:

The *"Mobile Meal Booking System"* should have the following modules:

1) An authentication and authorization module, which will handle the user request for login and account creation.

2) A meal manage module to perform tasks such as view booked meal, book meal, un-book meal, change diet and so on.

3) An information display module that gives user information about the status of any transaction, such as success reports or error messages.

4) A feedback module that allows users to contact the management and/or the administrator. In the case of a meal booking system email is the best choice for carrying out this task.

5) An online Help module that provides instructions on how to use the system.

Step 3: Abstract interactions between modules:

Figure 3.3.5-1 demonstrates the interactions and relationships between modules in a flow chart.

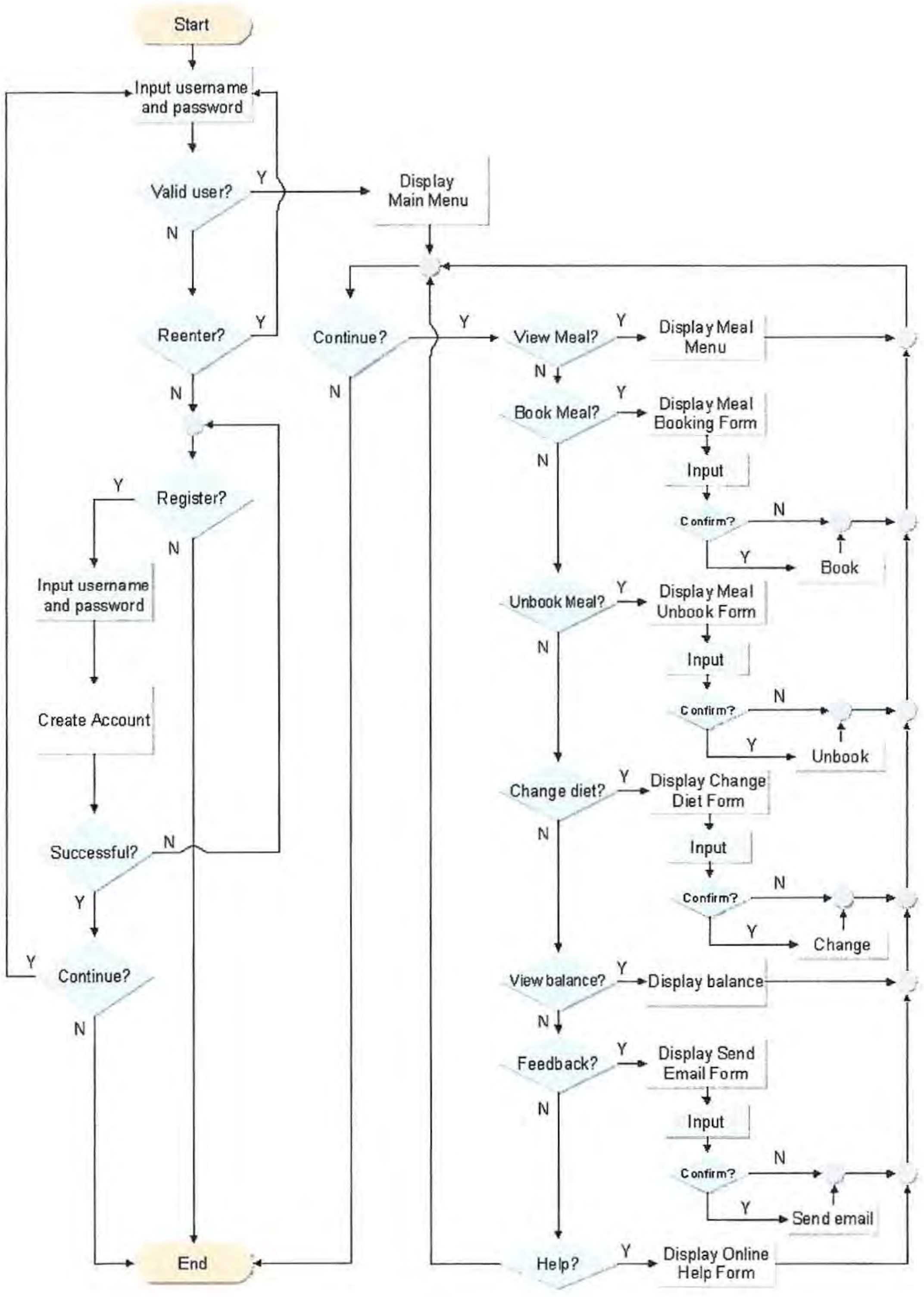


FIGURE 3.3.5-1 – MEALBOOKINGMIDLET LOGIC FLOW CHART

Step 4: Design user interfaces and GUI layout:

Using J2ME to design GUI components for mobile devices under MIDP is quite different from designing common Java applications. Common Java applications usually use AWT or swing[10] to design GUI components and they use the concept of layout and position. Therefore developers are used to specifying the actually physical presentation of Java applications by assigning them the layout or indicating the on screen position.

The layout/position approach works fine with normal Java applications because they are designed for personal computers, but not when developing Java mobile applications. As is well known, the display capability of mobile devices varies dramatically from one to another. Some devices can display only few lines of monochromic characters, while others might have a bigger screen and colour support. Therefore, it is impractical to design GUIs for a single device with a specific size and layout and expect the presentation to be of the same quality on other devices.

MIDP comes up with two new ideas to specify GUI in any MIDlet – Abstraction & Discovery:

1) Abstraction

Abstraction is a term used to specify GUIs conceptually, while the physical presentation is reliant on the MIDP implementation. For example, instead of specifying, "Put a 30mm text box at (x, y), where x and y present the screen coordinate", we say, "I need a text box somewhere on the screen".

2) Discovery

The discovery concept allows the MIDlet to learn about the mobile device capability at runtime and then render the GUI accordingly.

The GUI design of the "*Mobile Meal Booking System*" is more about logical functions than physical presentation. And the appearance sequence of GUI components is defended by using the concepts of abstraction and discovery. The GUI components of this application are designed to have proper representations on a wide range of Java-enabled mobile devices.

[10] Java Abstract Windows Toolkit (AWT) and Swing are Java packages that offer sets of classes and interfaces with which programmers canbuild Java Graphical User Interfaces.

3.3.5.2 STAGE TWO: Program the MealBookingMIDlet

Step 1: Coding

Coding a MIDlet is just like coding any other Java class, any text editor or IDE can be used to write MIDlet source code. SunOne Studio Mobile Edition is used to program the MealBookingMIDlet.

1) The MealBookingMIDlet class:

The MealBookingMIDlet class is driven from the base class javax.microedition.midlet.MIDlet. It can therefore use the application management software to control the running of the application, allow it to retrieve properties from the application descriptor and notify or request state changes.

2) The MealBookingMIDlet constructor:

The constructor is called once when the MealBookingMIDlet is loaded by the runtime environment.

3) The states transition methods:

As demonstrated in section 3.3.2 and figure 3.3.2-1, the base class MIDlet provides three abstract methods: "startApp", "pauseApp" and "destroyApp". These three methods should be implemented by the driven class –MealBookingMIDlet. After being implemented these three methods are used by the device's application management software to maintain the state changes of the application.

a) The "startApp()" Method:

The "startApp()" method is invoked immediately after the MealBookingMIDlet constructor and anytime the application is made active (not only when the system is initially launched). The application's state may change from being active to inactive many times during a single run. Therefore, to prevent the user interfaces from being initialised multiple times, the GUI initialisation method for the MealBookingMIDlet is put inside the class constructor, instead of in the "startApp()" method.

b) The "PauseApp()" Method:

Because most mobile devices lack the ability of real multi-processing, applications are often switched from one to another. The "pauseApp()" method indicates that the application is about to be paused. The "startApp()"

method is called by the application management software when the application is resumed.

c) The “destroyApp()” Method:

The “destroyApp()” method is called by the application management software to indicate that an application is about to be shut down. Unlike “startApp()” this method will be called only once during an application’s lifetime.

4) The states notification methods:

The MIDlet class provides a set of final methods that can be used to communicate with the application manager. Therefore the MIDlet can initiate some state changes itself and notify the application management software of those state changes by invoking the appropriate methods.

a) “notifyDestroyed()” method is used by a MIDlet to notify the application management software that it has entered into the “Destroyed” state.

b) “notifyPaused()” method notifies the application management software that the MIDlet does not need to be active and has entered the Paused state.

c) “resumeRequest()” method provides a MIDlet with a mechanism to indicate that it is interested in entering the Active state.

d) “getAppProperty()” method provides MIDlet with a mechanism to retrieve named properties from the application management software.

5) The functional methods:

Other methods defined inside the MealBookingMIDlet are functional methods, such as “splash” method, “login” method, “register” method, “BookingMenu” method, “bookMeal” method and so forth. These methods provide the actual logic and interface of the application.

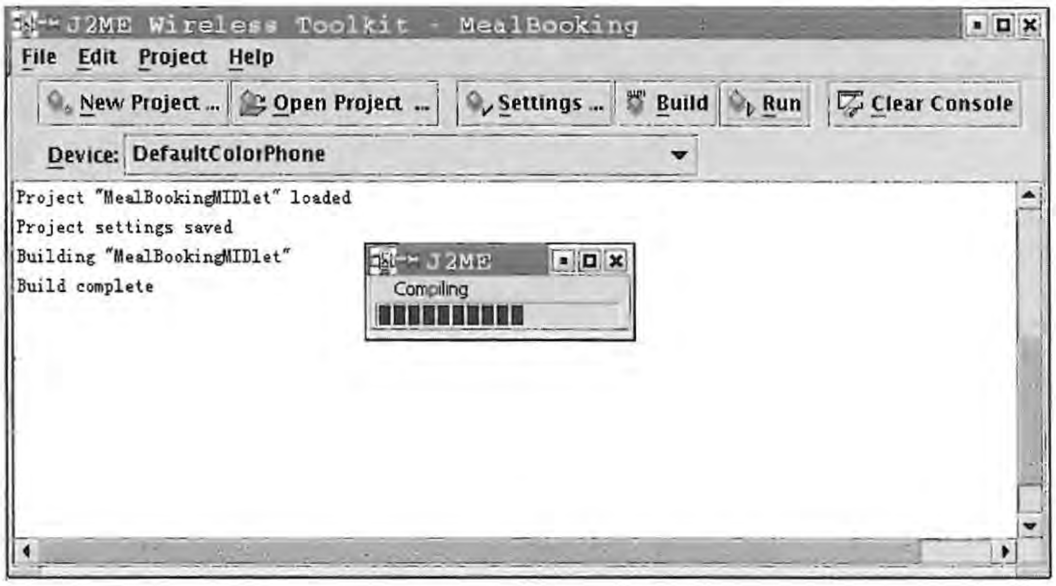
The following source code snippet [Code 3.3.5-1] shows the formation of the MealBookingMIDlet and the structure of the methods.

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;

public class MealBookingMIDlet extends MIDlet {
    ...
    public MealBookingMIDlet(){
        splash();
    }//constructor

    public void startApp(){
        ...
    }

    public void pauseApp() {
        ...
    }

    public void destroyApp(boolean unconditional) {
        cleanup();
    }//destroyApp
    ...
    private void splash(){

        final Form splashForm = new Form("");
        splashForm.addCommand(cmdExit);
        ...
        splashForm.setCommandListener(new CommandListener(){
                public void commandAction(Command cmd, Displayable dis){
                        notifyDestroyed();
                }
            }
        );
        ...
    }//splash()
    ...
    //other methods
    ...
}//MealBookingMIDlet class
```

CODE 3.3.5-1 – MEALBOOKINGMIDLET CLASS AND METHODS

Step 2: Compile the MealBookingMIDlet

The MealBookingMIDlet source code can be compiled in command-line mode just like any other java application. Alternatively we can use J2ME Wireless Toolkit – KToolbar - to compile. If there is any compile time error the error message will be displayed within the console.


J2ME Wireless Toolkit - MealBooking
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run Clear Console
J2ME
Compiling

Device: DefaultColorPhone

```
Project "MealBookingMIDlet" loaded
Project settings saved
Building "MealBookingMIDlet"
Build complete
```

FIGURE 3.3.5-2 – USING J2ME WIRELESS TOOLKIT

Step 3: Bytecode Preverfication

Preverification is a special feature provided by CLDC/MIDP specification. Unlike other java applications, which complete their bytecode verifications during the runtime, CLDC/MIDP splits bytecode verification for all MIDlets is into two steps. The first step is the bytecode preverification, which is performed during the development stage, right after compilation. The second is done on the mobile device during runtime. The mobile device is only required to do a "lightweight" second verification thereby reducing the runtime computation overhead.

After compiling the sources the development environment passes the generated class files to the Preverifier. This tool rearranges bytecodes in the classes to simplify the final stage of bytecode verification on the CLDC virtual machine. It also checks for the use of virtual machine features that are not supported by the CLDC [Please refer to figure 3.3.5-3 and figure 3.3.5-4].

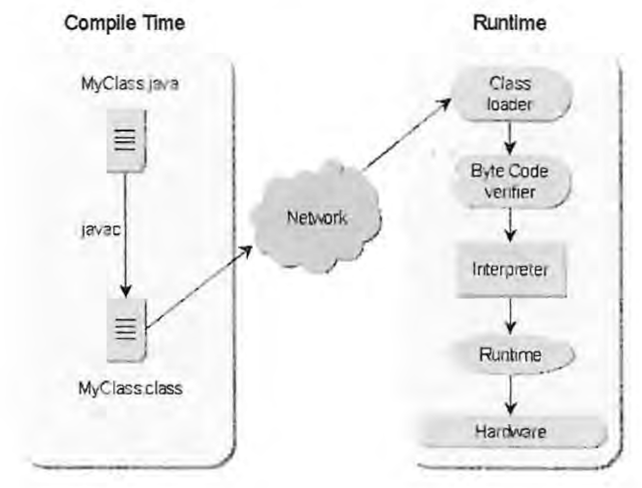


FIGURE 3.3.5-3 – J2SE RUNTIME ENVIRONMENT

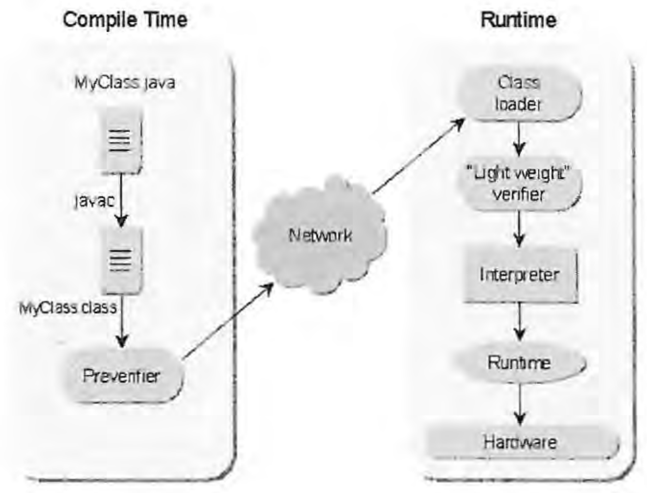


FIGURE 3.3.5-4 – J2ME - CLDC/MIDP RUNTIME ENVIRONMENT

3.3.5.3 STAGE THREE: Test the MealBookingMIDlet

J2ME mobile devices emulators:

Before deploying the MealBookingMIDlet onto any real mobile devices, emulators are used to test the application. Emulators are software simulations of hardware environments. In this case the hardware refers to mobile devices with certain functionalities and restrictions [Please see figure 3.3.5-5]. Emulating provides a flexible and economical way to test mobile application in an "almost real" environment. The SUN J2ME Wireless Toolkit supplies several configurations of different mobile device emulators, including default grey phone, default colour phone, minimum phone, Motorola i85s and RIM Java handheld. Other emulators such as Palm OS III can also be integrated with the Wireless Toolkit using a "plug-in like" manner.

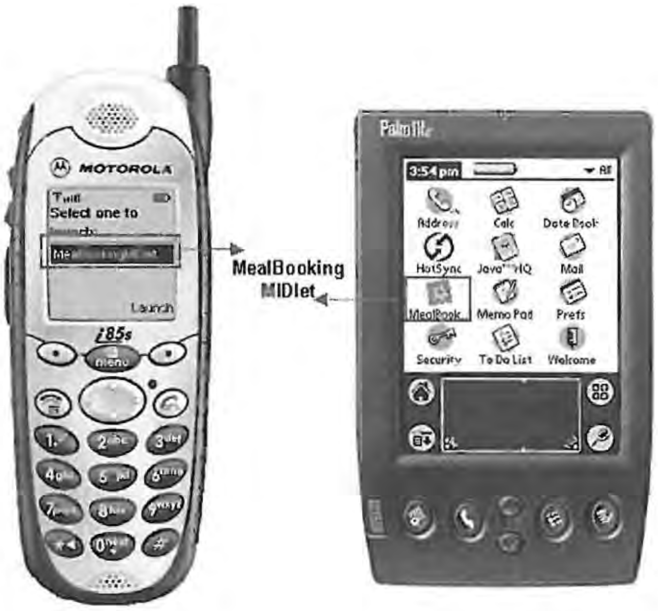


FIGURE 3.3.5-5 – MOTOROLA I85S & PALM OS III EMULATORS

For any standard J2ME-CLDC/MIDP enabled device, be that a Cell Phone or PDA, although the appearances and functions may vary from one to another, there are some major features in common:

1) Their physical interfaces are limited. The size of the screen is small and the input possibility and mechanisms are limited.

2) Two or more soft buttons are provided for the mobile device [Please see figure 3.3.5-6]. Soft buttons are buttons that do not have a pre-defined label and function, their value is assigned programmatically by applications during the runtime. This offers more flexibility for mobile applications.

3) Navigation buttons are available for users to scroll through lists or sets of choices.

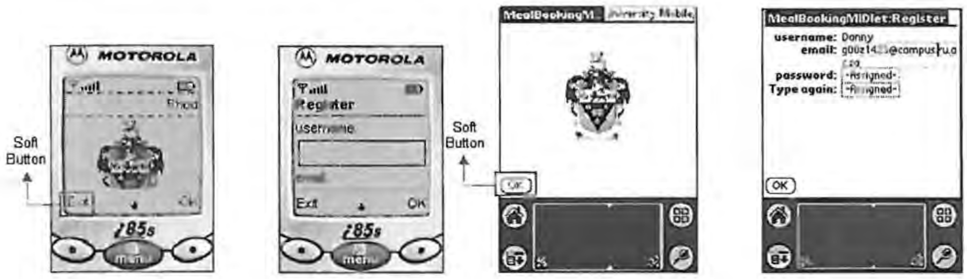


FIGURE 3.3.5-6 – RUNNING MEALBOOKINGMIDLET ON EMULATORS

"Real world" condition simulation:

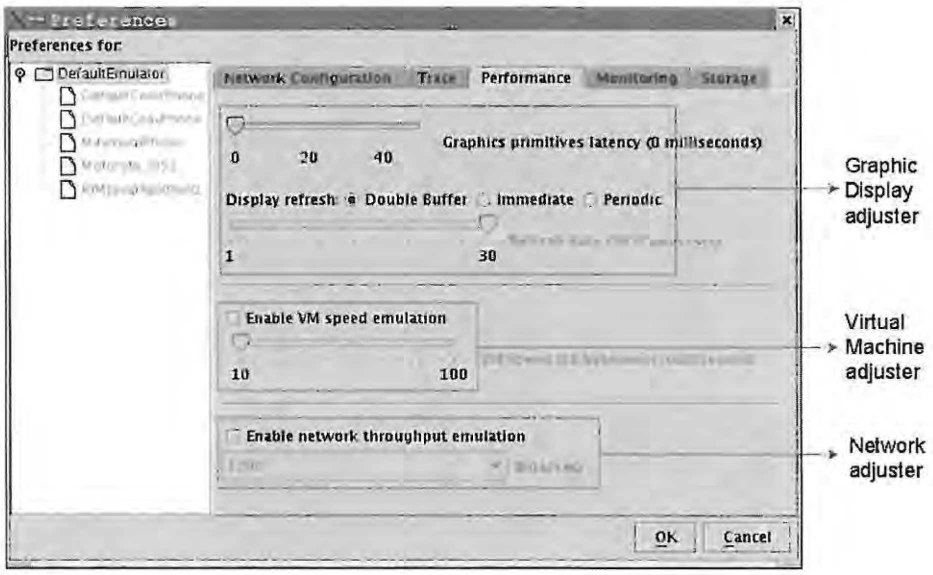
Testing the mobile Web application on an emulator usually gives a satisfactory performance and result. This is because the emulator runs on a PC that has great computing power, sufficient memory and network bandwidth. In the real world things are different though. Mobile devices often suffer from a shortage of memory or an unstable network transmission. In order to get more realistic results from an emulator the following factors must be considered and simulated.

1) The time required to draw the GUI on the screen (The drawing speed and refreshing rate).

2) The speed of the device on which applications are running. (Virtual machine speed and device processing speed)

3) The network throughput and transferring speed.

The J2ME Wireless Toolkit offers a utility to simulate these facts [Please see figure 3.3.5-7]. By scaling down on the performance of a selection of subsystems of the emulator, adjusting the speed of on-screen drawing, virtual machine speed and the network throughput, developers can optimise the overall performance of applications for both high-end and low-end devices.


Preferences for: DefaultEmulator

Graphics primitives latency (0 milliseconds)

Display refresh: [✓ Double Buffer] Immediate Periodic

Enable VM speed emulation

Enable network throughput emulation

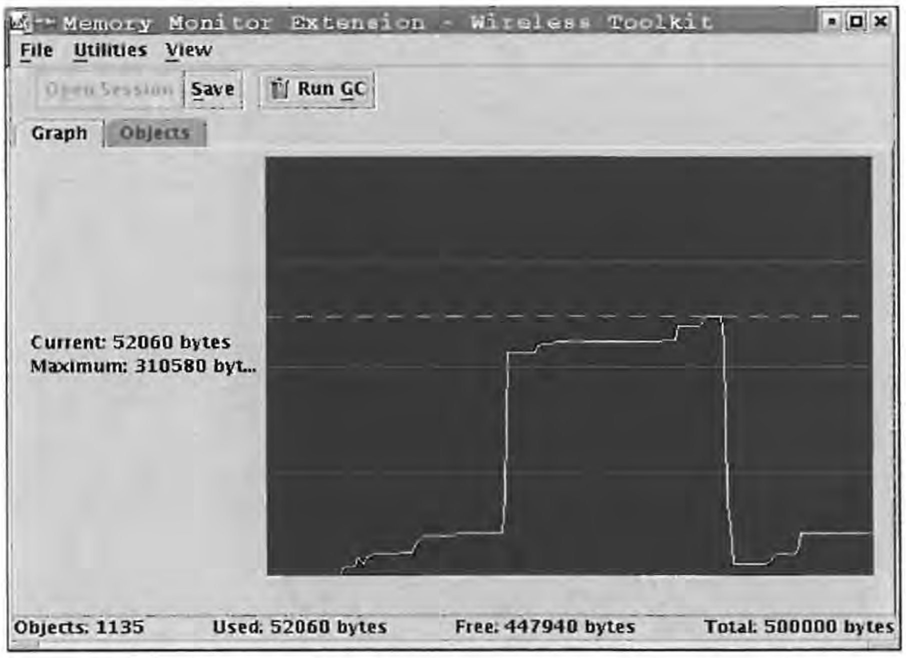
FIGURE 3.3.5-7 – WIRELESS TOOLKIT PERFORMANCE ADJUSTING UTILITY

Mobile Web application monitoring and profiling:

J2ME Wireless Toolkit supplies developers with a set of monitoring and profiling tools to examine a range of aspects that are critical to the application performance during runtime. These include network connectivity, memory usage and method execution time. Charts and statistics are generated based on the sampled figures.

1) The memory usage monitor:

By using the monitor the application memory usage is exposed during runtime, so that developers can determine where a bottleneck is affecting an application's performance. The memory usage monitor provides a variety of information on memory and objects, which includes: the initial amount of memory available, the current amount of memory used by the application, the maximum amount of memory used since the last program execution, the current amount of free memory available, the total number of class objects allocated at start-up, the total amount of memory used by the class's live objects, the average amount of memory used by a class live object with respect to the total size, the number of objects in the heap, the name of each class examined, and the number of instances of an object in the heap [Please see figure 3.3.5-8].



Current: 52060 bytes
Maximum: 310580 byt...
Objects: 1135 Used: 52060 bytes Free: 447940 bytes Total: 500000 bytes

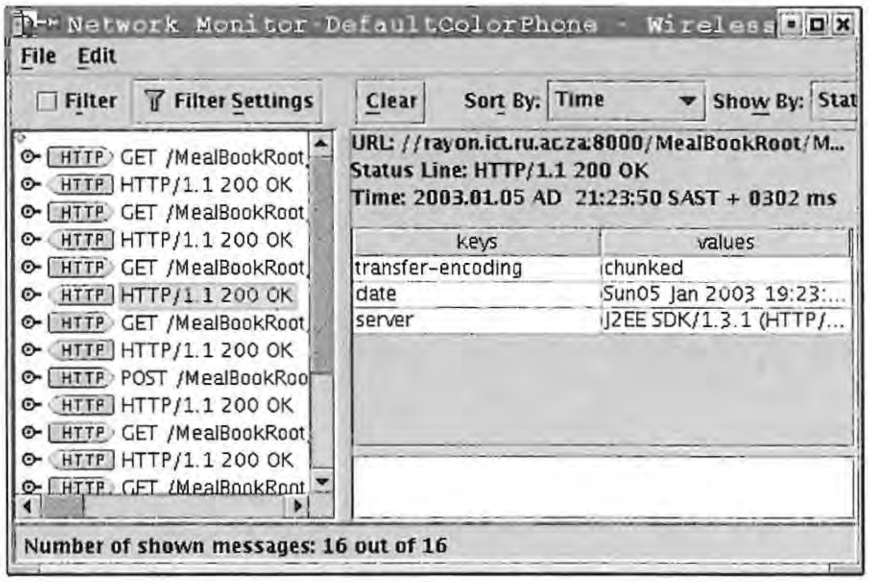
FIGURE 3.3.5-8 – MEMORY USAGE MONITOR

2) The Network activity monitor:

As a mobile Web application, the MealBookingMIDlet relies heavily on the Internet to communicate with the Web component through HTTP. With the network activity monitor information about the network status of the message transferred can be obtained, bottlenecks can be detected and bugs can be fixed. In this way network usage of the application can be optimised.

The network activity monitor supports both HTTP transmission (for common Internet access, which is used for the MealBookingMIDlet) and HTTPS transmission (for security Internet communication). It also supports Internet access via proxy servers.

The network activity monitor intercepts and displays a list of information that was sent or received by the application. It contains the accessing URL, protocol type and version, the date and time the message was sent or received, the properties of the message and even the full message body [Please see figure 3.3.5-9].



```
HTTP GET /MealBookRoot
HTTP HTTP/1.1 200 OK
HTTP GET /MealBookRoot
HTTP HTTP/1.1 200 OK
HTTP GET /MealBookRoot
HTTP HTTP/1.1 200 OK
HTTP GET /MealBookRoot
HTTP HTTP/1.1 200 OK
HTTP POST /MealBookRoo
HTTP HTTP/1.1 200 OK
HTTP GET /MealBookRoot
HTTP HTTP/1.1 200 OK
HTTP GET /MealBookRoot
```

URL: //rayon.ict.ru.ac.za:8000/MealBookRoot/M...
Status Line: HTTP/1.1 200 OK
Time: 2003.01.05 AD 21:23:50 SAST + 0302 ms

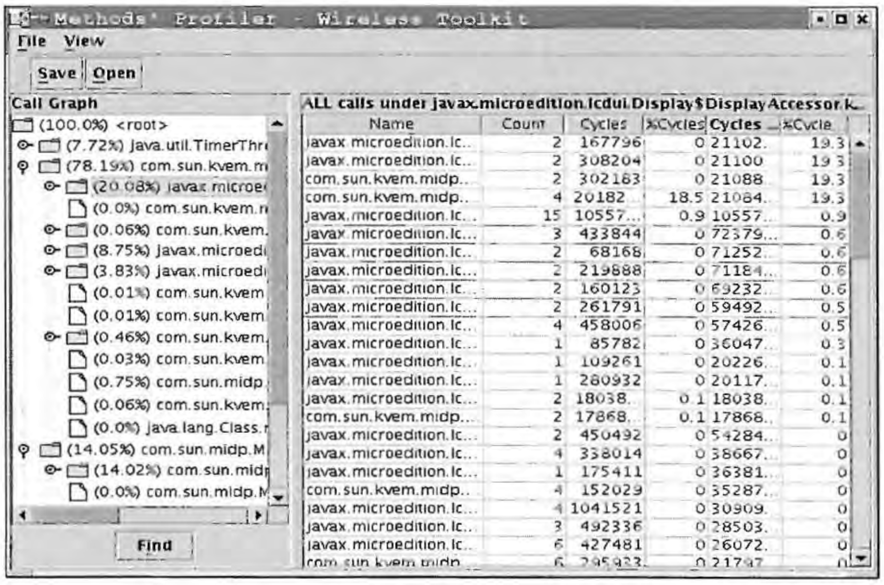
keys	values
transfer-encoding	chunked
date	Sun05 Jan 2003 19:23:...
server	J2EE SDK/1.3.1 (HTTP/...

Number of shown messages: 16 out of 16

FIGURE 3.3.5-9 – NETWORK USAGE MONITOR

3) Methods' profiler:

The methods' profiler gathers data at runtime from the emulator and records the time of each method execution. The Profiler window displays the method information in two ways: a hierarchical view of method relationships and execution time (in both amount and percentage) and the number of times a method and its descendants were called during runtime.



Call Graph

```
(100.0%) <root>
  (7.72%) java.util.TimerThr
  (78.19%) com.sun.kvem.m
    (20.08%) javax.microe
      (0.0%) com.sun.kvem.r
    (0.06%) com.sun.kvem.
    (8.75%) javax.microedi
    (3.83%) javax.microedi
      (0.01%) com.sun.kvem
      (0.01%) com.sun.kvem
    (0.46%) com.sun.kvem.
      (0.03%) com.sun.kvem
      (0.75%) com.sun.midp
      (0.06%) com.sun.kvem.
      (0.0%) java.lang.Class.r
  (14.05%) com.sun.midp.M
    (14.02%) com.sun.midp
      (0.0%) com.sun.midp.M
```

ALL calls under javax.microedition.lcdui.Display$DisplayAccessor.k

Name	Count	Cycles	%Cycles	Cycles ...	%Cycle
javax.microedition.lc...	2	167796	0	21102.	19.3
javax.microedition.lc...	2	308204	0	21100	19.3
com.sun.kvem.midp...	2	302183	0	21088	19.3
com.sun.kvem.midp...	4	20182	18.5	21084.	19.3
javax.microedition.lc...	15	10557...	0.9	10557...	0.9
javax.microedition.lc...	3	433844	0	72379...	0.6
javax.microedition.lc...	2	68168	0	71252.	0.6
javax.microedition.lc...	2	219888	0	71184...	0.6
javax.microedition.lc...	2	160123	0	69232.	0.6
javax.microedition.lc...	2	261791	0	59492...	0.5
javax.microedition.lc...	4	458006	0	57426.	0.5
javax.microedition.lc...	1	85782	0	36047.	0.3
javax.microedition.lc...	1	109261	0	20226.	0.1
javax.microedition.lc...	1	280932	0	20117.	0.1
javax.microedition.lc...	2	18038.	0.1	18038.	0.1
com.sun.kvem.midp...	2	17868.	0.1	17868.	0.1
javax.microedition.lc...	2	450492	0	54284...	0
javax.microedition.lc...	4	338014	0	38667...	0
javax.microedition.lc...	1	175411	0	36381.	0
com.sun.kvem.midp...	4	152029	0	35287...	0
javax.microedition.lc...	4	1041521	0	30909.	0
javax.microedition.lc...	3	492336	0	28503.	0
javax.microedition.lc...	6	427481	0	26072.	0
com.sun.kvem.midp...	6	295923.	0	21797	0

FIGURE 3.3.5-10 – THE METHODS' PROFILER

3.3.5.4 STAGE FOUR: Packing and deploying MealBookingMIDlet

Packing the MealBookingMIDlet:

As shown in figure 3.3.5-11, the MealBookingMIDlet and its resources, images, icons etc., must be packaged into a MIDlet-suite before deploying to any real devices. Packaging a MIDlet manually is a fairly complex process. A manifest file should be created first, then the Java Archive (JAR) file, and finally, a Java Application Descriptor (JAD) file.

```
MIDlet-1: MealBookingMIDlet, MealBookingMIDlet.png, MealBookingMIDlet
MIDlet-Jar-Size: 32317
MIDlet-Jar-URL: MealBookingMIDlet.jar
MIDlet-Name: MealBookingMIDlet
MIDlet-Vendor: Sun Microsystems
MIDlet-Version: 1.0
```

CODE 3.3.5-2 – MANIFEST.MF FILE

```
MIDlet-1: MealBookingMIDlet, MealBookingMIDlet.png, MealBookingMIDlet
MIDlet-Name: MealBookingMIDlet
MIDlet-Vendor: Sun Microsystems
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-1.0
```

CODE 3.3.5-3 – MEALBOOKINGMIDLET.JAD FILE

Fortunately the Wireless Toolkit provides a tool to generate and package the MIDlet-suit automatically – a packaging tool that supports bytecode obfuscation.

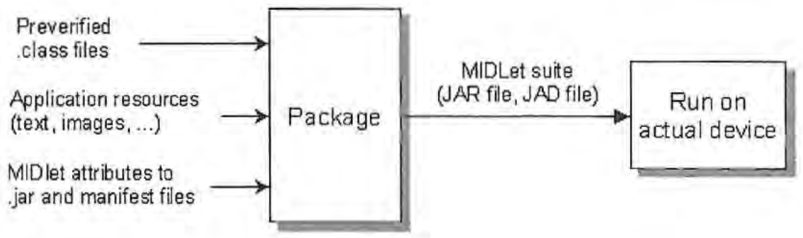

Preverified .class files
Application resources (text, images, ...)
MIDlet attributes to .jar and manifest files
Package
MIDLet suite (JAR file, JAD file)
Run on actual device

FIGURE 3.3.5-11 – PACKAGING MIDLET

Deploying the MealBookingMIDlet:

Once the MealBookingMIDlet is successfully packaged it is time to port it onto the destination device. There are two ways to install a MIDlet. If the mobile device is connected to a PC via USB or another connection the MIDlet can be downloaded from the Internet to the PC and by using synchronization software on the PC, which usually comes with the mobile device, one can install the MIDlet onto the mobile device. Otherwise if the mobile device has a direct connection to the Internet the MIDlet can be downloaded and installed directly [See figure 3.3.5-12].

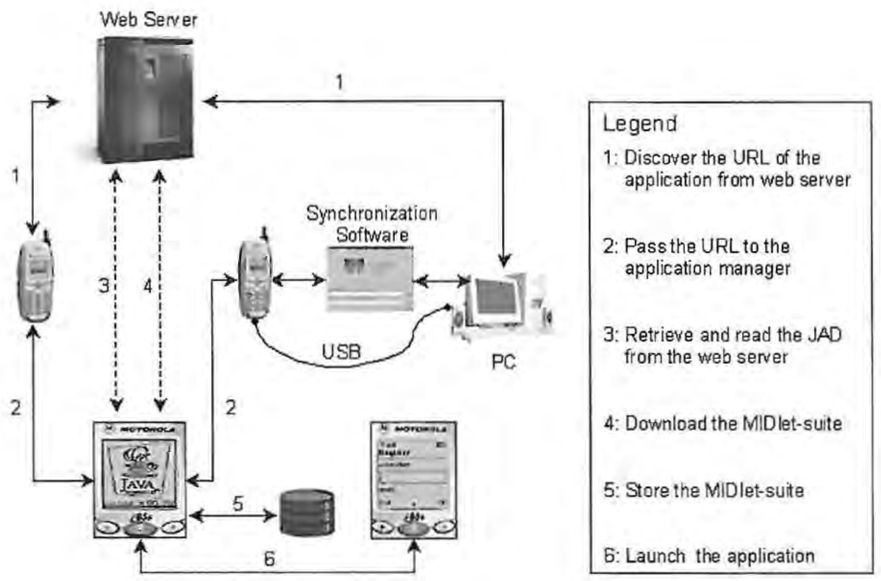

Web Server
1
1
2
3
4
2
Synchronization Software
USB
PC
5
6
Legend
1: Discover the URL of the application from web server
2: Pass the URL to the application manager
3: Retrieve and read the JAD from the web server
4: Download the MIDlet-suite
5: Store the MIDlet-suite
6: Launch the application

FIGURE 3.3.5-12 – DEPLOYING MIDLET ONTO MOBILE DEVICE

3.4 Developing the middle tier Web component

After the above detailed introduction of the MealBookingMIDlet, let us take a look at the development and deployment of the middle tier Web component – the MealBookingServlet.

There are a number of server-side technologies in existence today, such as the Common Gateway Interface (CGI), Active Server Pages (ASP) and Java Server Pages (JSP). Java Servlet technology is used to implement the Web component for the *"Rhodes University Mobile Meal Booking System"*.

3.4.1 Servlet – the "Non-Visual Applet" on the server-side

3.4.1.1 Java Servlet

"A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model. Although Servlets can respond to any type of request, they are commonly used to extend the applications hosted by Web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes. The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets [Please see figure 3.4.1-1]. All servlets must implement the Servlet interface, which defines life-cycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services."[Sun Microsystems]

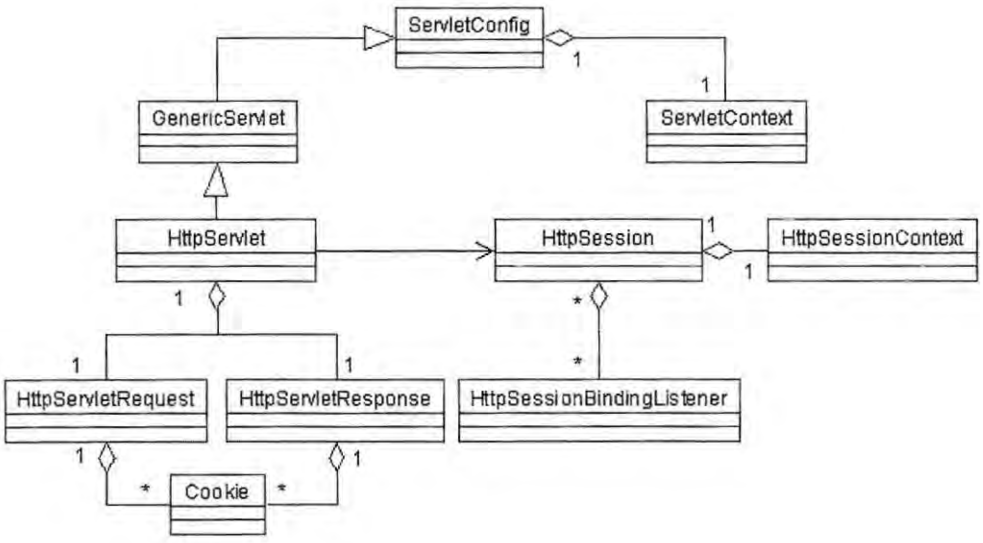


FIGURE 3.4.1-1 – JAVA SERVLET API

3.4.1.2 Servlet life cycle

The life cycle of a servlet is controlled by the Web container[11] in which the servlet has been deployed. When the Web container receives a request to a servlet, and if there is no instance of the requested servlet, the Web container loads the servlet class and creates an instance of the servlet class. It initialises the servlet instance by calling the "init()" method. If an instance of the servlet does exist the Web container invokes the "service()" method, passing it a request and response object. Otherwise if the container needs to remove the servlet it finalizes the servlet by calling the servlet's "destroy()" method.

3.4.1.3 HttpServlet

HttpServlet is an abstract class that extends the GenericServlet class and provides an HTTP specific implementation of the Servlet interface. It is the most commonly used base-class for servlet implementations. The MealBookingServlet extends the HttpServlet and overrides some of its methods to handle requests to and from the MealBookingMIDlet.

HttpServlet Methods	
Methods	**Descriptions**
DoDelete()	Allow a servlet to handle a DELETE request.
doGet()	Allow a servlet to handle a GET request.
doHead()	Receives and handles an HTTP HEAD request
doOptions()	Allow a servlet to handle an OPTIONS request.
doPost()	Allow a servlet to handle a POST request.
doPut()	Allow a servlet to handle a PUT request.
doTrace()	Allow a servlet to handle a TRACE request.
getLastModified()	Returns the time the HttpServletRequest object was last modified
service()	Receives standard HTTP requests and dispatches them to the other methods defined in this class

TABLE 3.4.1-1 – HTTPSERVLET METHODS

3.4.2 HTTP requests

In order to interact with the Web component and make it run a server-side program the MealBookingMIDlet needs to issue a request to the MealBookingServlet. There are three HTTP request methods, namely HTTP HEAD, HTTP GET and HTTP POST.

[11] Container is a J2EE architecture concept. Containers are the interface between a component and the low-level platform-specific functionality that supports the component. Before a Web, enterprise bean, or application client component can be executed, it must be assembled into a J2EE application and deployed into its container.

3.4.2.1 HTTP HEAD request

The HTTP HEAD request retrieves only the header – descriptive information – of a document on the Web server.

3.4.2.2 HTTP GET request

HTTP GET is used to send requests to the Web server and ask for execution of some Server-side programs. The data being passed to the Web server is added behind the URL string as "key=value" pairs in plain text. In the MealBookingServlet the "doGet()" method is used to deal with meal booking requests from the MealBookingMIDlet.

3.4.2.3 HTTP POST request

HTTP POST is similar to the HTTP GET method. They basically complete the same task in a different manner. Instead of appending the request data to the URL, HTTP POST encapsulates the information into the request body and sends it to the Web server. Obviously it is more secure than the HTTP GET method. "doPost()" method is used for handling login, registration, and the send-email request from the MealBookingMIDlet.

3.4.3 Servlet session

3.4.3.1 HTTP session

There are two different types of Internet communication protocols, one is stateless protocols, the other is stateful protocols. Stateless protocols do not keep a record of states of connections made between client and server. Stateful, on the contrary, do keep such a record. These connections are called sessions. "Formally a session may be defined as an interaction between client and server where the server can associate a number of requests with a particular client over defined time scale" [Poulton, Austin]. Originally HTTP was designed as a simple stateless protocol for electronic documents being shared among anonymous users throughout the Internet. Therefore the Web server is not capable of identifying specific client information amongst multiple requests. This works well for the traditional Web content, but for most Web applications, including our *"Mobile Meal Booking System"*, access to state information is critical.

3.4.3.2 HTTP session tracking

The technique of maintaining the state information between server round trips is called session tracking. Essentially, session tracking is to pass data generated from one request on to subsequent requests.

To date several techniques of session tracking have been developed to fulfil the growing demand of maintaining state information for Web applications. These techniques include hidden form fields, HTTP user authorization, cookies, and URL rewriting.

Hidden form field:

Hidden form field is a fairly simple session tracking technique. As indicated by the name, it keeps the session information by writing it into the value of hidden fields on a Web form.

HTTP user authorization:

The "HTTP user authorization" technique supports session tracking via the HTTP "User Authorization" response header. The user is asked to provide his username and password when he/she requests the connection for the first time. Thereafter that username can be obtained from the "getRemoteUser()" method.

Cookies:

Cookies are text files stored in the client browser, which keep the state information by writing into and reading from those local files.

URL rewriting:

The *"Mobile Meal Booking System"* uses the URL rewriting technique to maintain session states. This approach sends the states information to the Web server by adding "key=value" pairs after each HTTP GET request [please refer to code 3.4.3-1 for an example]. The Servlet can therefore retrieve the information via the Request object [See code 3.4.3-2].

```
String URL = "http://rayon.ict.ru.ac.za:8000/MealBookRoot/MealBookServlet";
//Rewrite the URL
URL += "?type=bookMeal&meal=" + parameter[0] +
          "&mealType=" + parameter[1] +
          "&day="+ parameter[2] +
          "&bookFrom=" + parameter[3] +
          "&bookTo="+ parameter[4] +
          "&username=" + user;
//establishing connection to the http server by sending the rewrote URL
connection = (HttpConnection)Connector.open(URL);
```

CODE 3.4.3-1 – REWRITING AND SENDING URL VIA HTTP GET REQUEST (CLIENT)

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    ...
        String meal = request.getParameter("meal");
        String mealType = request.getParameter("mealType");
    ...
}
...
```

CODE 3.4.3-2 – PARSING THE REQUEST AND GETTING THE INFORMATION (SERVER)

3.4.4 Servlet database connectivity

Database access and operation is essential to most Web applications, and all modern server-side technologies have connectivity mechanisms for a wide array of databases. The Servlet is no exception.

3.4.4.1 JDBC & JDBC Driver

JDBC stands for Java Database Connectivity and is an API that is defined as part of Java 2 Standard Edition. It provides a set of interfaces and classes to perform database-related operations. JDBC abstracts database specific details and generalizes the most common database access functions.

JDBC driver is what the Java Virtual Machine uses to translate the generalized JDBC calls into vendor specific database calls [Please see figure 3.4.4-1]. The drivers are java classes loaded at run time. This allows for flexibility when deploying applications that access databases from multiple vendors.

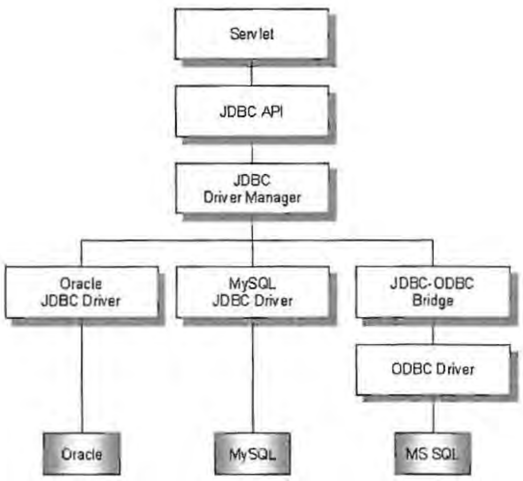


FIGURE 3.4.4-2 – JDBC INFRASTRUCTURE

For the Web component of the *"Mobile Meal Booking System"* the "MealBookingServlet" uses "JDBC-MySQL Driver" to set up the connection and access the MySQL database. The java.sql package from Java 2 Standard Edition provides a collection of interfaces and classes, such as DriverManager, SQLData, SQLInput, to carry out various database operations via the JDBC-MySQL driver. Code 3.4.4-1 is an example on how to register the JDBC-MySQL driver and perform database operations.

```
...
private final static String DRIVER ="com.mysql.jdbc.Driver";
private Statement stmt;
...
public void init() throws ServletException{

    try{
        //create an instance and register the JDBC-MySQL driver
        Class.forName(DRIVER).newInstance();
        stmt = con.createStatement();
        ...
    }
    catch(Exception e){
        ...
    }
}
...
public ResultSet queryDB(String query){

    try{
    //Execute query on the MySQL database
        return stmt.executeQuery(query);
    }
    catch(SQLException e){
        e.printStackTrace();
        return null;
    }
}
...
```

CODE 3.4.4-1 – SERVLET DATABASE CONNECTIVITY USING JDBC

3.4.5 Developing Servlet for the "Mobile Meal Booking System"

The introduction to the development of the MealBookingServlet will be presented in the same way as for the MealBookingMIDlet, which follows the steps of:

designing → programming → testing → deploying.

3.4.5.1 STAGE ONE: Design the MealBookingServlet

Step 1: Define design goals

1) The MealBookingServlet should be able to run on any Servlet-aware Web server.

2) The MealBookingServlet should listening to port 8000, capturing any incoming HTTP requests, calling methods based on the request type and content and sending responses back to the client as a result.

3) The MealBookingServlet should be able to interact with a backend database and perform operations such as query, update and delete.

4) The MealBookingServlet should be able to handle the send-email request from the client and set up the connection with the SMTP (Simple Message Transfer Protocol) server.

5) The MealBookingServlet should be stable, capable of recovering from a SQL error a SMTP error, and provide error messages to the client accordingly.

Step 2: Categorize function modules

The function modules in the MealBookingServlet can be mapped with a one to one relationship with methods in the MealBookingMIDlet. Methods in MealBookingMIDlet supply data and issue requests for operations. Methods in MealBookingServlet perform the actual operations on the backend database and provide the results accordingly.

1) An authentication and authorization module that takes the parameter from the client request, queries the database for authentication and sends the results back to client. If, for example, the request is for registration, it creates a new account in the database.

2) A meal-managing module that queries and updates the meal-related records in the database according to the client requests.

3) An email module that performs the send-email task for the client.

Step 3: Abstract interactions between modules:

The following flow chart [Figure 3.4.5-1] illustrates the interactions and relationships between modules.

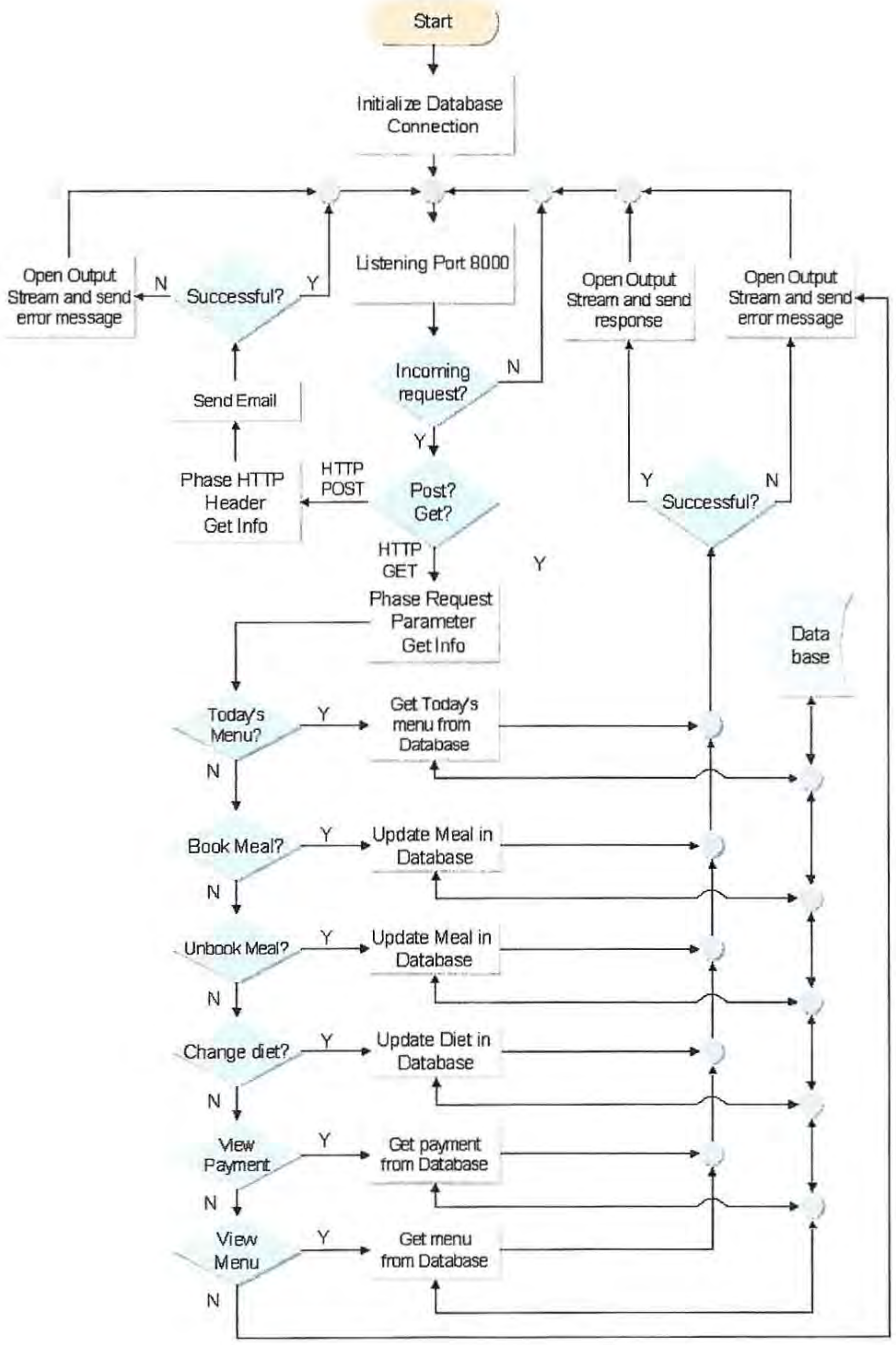


FIGURE 3.4.5-1 – MEALBOOKINGSERVLET LOGIC FLOW CHART

3.4.5.2 STAGE TWO: Program the MealBookingServlet

Step 1: Coding:

1) The MealBookingServlet class:

As described in section 3.4.1.3 and table 3.4.1-1 the MealBookingServlet class is driven from the abstract class javax.servlet.http.HttpServlet. It implements the "init()" method and overrides "doGet()" and "doPost()" methods. Other methods inherited from the HttpServlet class are not used [Please see code 3.4.5-1].

a) The "init()" method:

Inside the "init()" method the JDBC-MySQL Driver is registered, SQL statements are initialised and a database connection is established.

b) The "doGet()" method:

HTTP GET requests are dispatched and handled by the "doGet()" method. Request parameters are parsed to determine which functional method to call.

c) The "doPost()" method:

Within the MealBookingServlet class the "doPost()" method deals with the HTTP POST requests, parses the message body, gets the username, email content and calls the "sendMail()" method.

d) The functional methods:

Other methods defined inside the MealBookingServlet are functional methods, such as "login" method, "register" method and "bookMeal" method. These methods perform the actual operations on the database and provide the feedback.

2) The MailClient class:

The MailClient is a helper class that performs the task of sending email via a SMTP server. JavaMail API is used to establish the connection. The MailClient class contains a single public method, "sendMail()", which is called within the MealBookingServlet class to send the email [See code 3.4.5-2].

```
import java.sql.*;
import javax.servlet.http.*;
...
public class MealBookingServlet extends HttpServlet {
    ...
    public void init() throws ServletException{
        try{
        ...
            stmt = con.createStatement();
        }
        catch(Exception e){...}
    }//init

    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ...{
        ...
        if(request.getParameter("type").equals("login")){
            ...
            login(...);
        }
        if(request.getParameter("type").equals("bookMeal")){
            bookMeal(...);
        }
    }//doGet

    public void doPost(HttpServletRequest req, HttpServletResponse resp) throws ... {
        ...
        String content = request.getHeader("content");
        ...
        query = "SELECT email FROM accounts WHERE name="+"\""+username+"\"";
        ...
        if(new MailClient(emailFrom, content).sendMail()){
        ...
        }
    }//doPost

    private ResultSet queryDB(String query){...}

    private void login(...){...}

    private void bookMeal(...){...}
    ...
}//MealBookingServlet
```

CODE 3.4.5-1 – THE MEALBOOKINGSERVLET CLASS

```
import javax.mail.*;
import javax.mail.internet.*;
import javax.naming.*;
import java.io.*;

public class MailClient {

    private MimeMessage message;
    private String msgTo = "g00z1423@campus.ru.ac.za";
    private String msgSubject = "Meal booking issue";
    private String msgFrom;
    private String msgBody;

    public MailClient(String msgFrom, String msgBody){
        this.msgFrom = msgFrom;
        this.msgBody = msgBody;
    }//constructor

    public boolean sendMail(){
        try{
            Context initial = new InitialContext();
            Session session = (Session)initial.lookup("java:comp/env/TheMailSession");
            message = new MimeMessage(session);

            Address fromAddress = new InternetAddress(msgFrom);
            Address[] toAddress = InternetAddress.parse(msgTo);

            message.setFrom(fromAddress);
            message.setRecipients(Message.RecipientType.TO, toAddress);
            message.setSubject(msgSubject);
            message.setText(msgBody.toString());

            Transport.send(message);
            return true;
        }
        catch(Exception e){
            e.printStackTrace();
            return false;
        }
    }//sendMail
}//MailClient class
```

CODE 3.4.5-2 – THE MAILCLIENT CLASS

Step 2: Compiling and Packaging

1) Compiling:

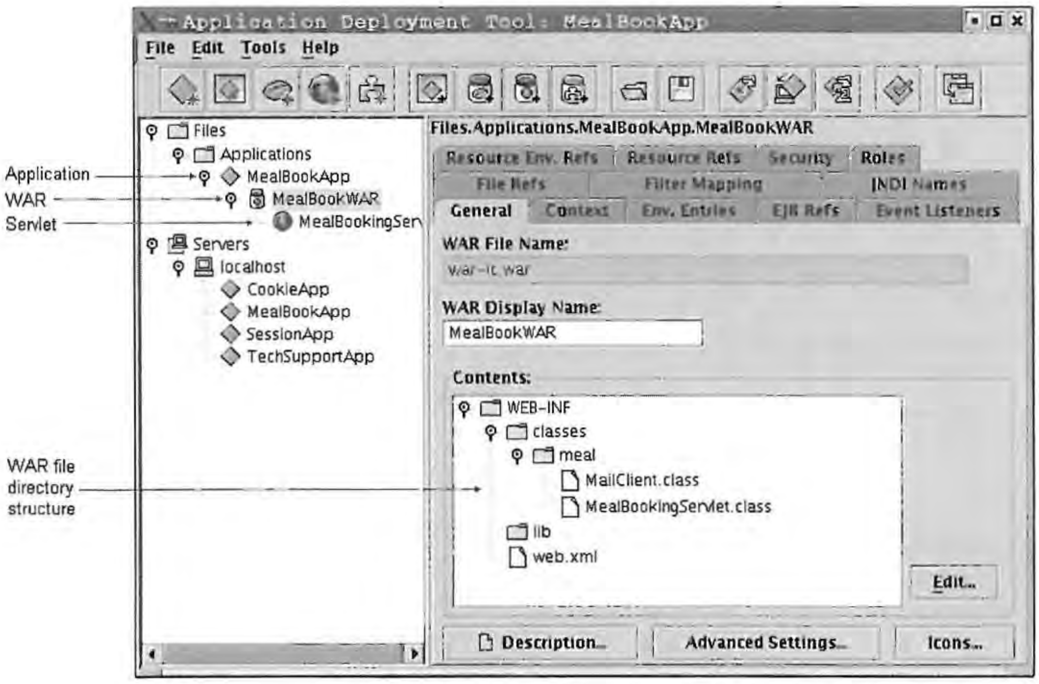
The MealBookingServlet and MailClient class are compiled in the command-line using the "javac" complier with JDBC-MySQL driver and J2EE class libraries.

```
javac –classpath %CLASSPATH% -d \dev\classes MealBookingServlet.java
```

CODE 3.4.5-3 – COMPILING MEALBOOKINGSERVLET CLASS

2) Packaging:

Once the classes are compiled they need to be packaged into a Web component archive (WAR) file – MealBookWAR. The MealBookWAR file has a specific directory structure when it is deployed. The top-level directory of the MealBookWAR is the root of the application. The root is where the Web resources are stored, such as HTML pages and JSP pages. There is a subdirectory within the root called WEB-INF, which contains "Web.xml" – the Web application deployment descriptor – as well as the class library directory that contains servlets, helper classes, and so on. J2EE Application Deployment Tool is used to create the MealBookWAR file.



Files.Applications.MealBookApp.MealBookWAR

WAR File Name: war-ic.war

WAR Display Name: MealBookWAR

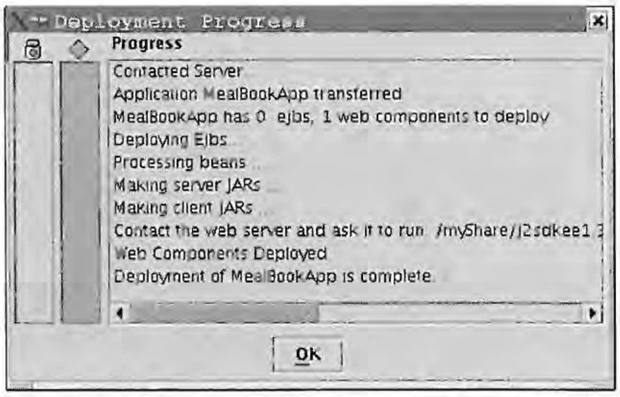
Contents:

```
WEB-INF
  classes
    meal
      MailClient.class
      MealBookingServlet.class
  lib
  web.xml
```

FIGURE 3.4.5-2 – J2EE APPLICATION DEPLOYMENT TOOL

Step 3: Testing and Deployment

Using the J2EE Application Deployment Tool the MealBookingServlet can be deployed as a WAR file in the J2EE Web application container onto the J2EE server and the Apache Web Sever.



```
Contacted Server
Application MealBookApp transferred
MealBookApp has 0  ejbs, 1 web components to deploy
Deploying Ejbs
Processing beans
Making server JARs
Making client JARs
Contact the web server and ask it to run  /myShare/j2sdkee1 3
Web Components Deployed
Deployment of MealBookApp is complete.
```

FIGURE 3.4.5-3 – DEPLOYING THE MEALBOOKINGSERVLET

To test it we can either use a Web browser or use the client-side application – the MealBookingMIDlet [Please refer to section 3.3.5.3].

3.5 Design the Back-end data component

The MySQL server serves as the backend database for the *"Mobile Meal Booking System"* [Please refer to section 3.2.1.1 for an introduction of the MySQL database].

3.5.1 MySQL command

MySQL provides a set of prompt-based utilities for maintenance and manipulation of the server and databases. Basically there are two categories of commands used to operate the MySQL database.

3.5.1.1 Under shell prompt:

```
a) Start MySQL server:
    Enter super user mode and then type the following command
    bash-2.05# safe_mysqld --user=mysql &
b) Stop MySQL:
    bash-2.05# mysql.server stop
c) Enter MySQL console:
    bash-2.05# mysql
```

CODE 3.5.1-1 – MYSQL COMMAND – SHELL PROMPT

3.5.1.2 Under MySQL prompt:

```
a) List all databases running on the MySQL server:
      mysql> show databases;
b) Change and use a database:
      mysql> use mealbooking;
c) List all tables in the current database:
      mysql> show tables;
d) Show table information, field name, data type etc.:
      mysql> describe mealmenu;
e) Query database:
      mysql> SELECT * FROM mealmenu;
f) Quit MySQL console and go back to shell prompt:
      mysql> quit
```

CODE 3.5.1-2 – MYSQL COMMAND – MYSQL PROMPT

3.5.2 Mealbooking database

The database – "mealbooking" – is created for the application It contains three tables, which are described as follows:

3.5.2.1 The "account" table:

The "account" table contains user information such as username, password, email address and account balance.

The "account" table is queried when the "login()" and "viewPayment()" methods are invoked and is updated when the "register()" method is called.

Accounts					
Field	**Type**	**Null**	**Key**	**Default**	**Extra**
Accounted	varchar(6)	NO	YES	NULL	
Name	varchar(8)	NO		NULL	
Password	varchar(6)	NO		NULL	
Email	varchar(30)	YES		NULL	
Defaultmeal	char(1)	NO		'N'	
Payment	decimal(10,2)	YES		NULL	

TABLE 3.5.2-1 – ACCOUNTS TABLE

3.5.2.2 The "mealmenu" table:

The "mealmenu" table holds a detailed description of daily meals. There are four types of meals available: Normal ('N'), Fast-food ('F'), Healthy ('H'), and Chinese ('C').

The "mealmenu" table is a read-only table. The application cannot modify it. It is queried when the "todayMenu()" and "viewMenu()" methods are invoked.

Mealmenu					
Field	**Type**	**Null**	**Key**	**Default**	**Extra**
MealID	varchar(10)	NO	YES	NULL	
Mealdate	Date	NO		NULL	
Mealtype	varchar(10)	NO		NULL	
Breakfast	varchar(60)	NO		NULL	
Lunch	varchar(60)	NO		NULL	
Supper	varchar(60)	NO		NULL	

TABLE 3.5.2-2 – MEALMENU TABLE

3.5.2.3 The "bookedmeal" table:

The "bookedmeal" contains information on meal types and meal status for each meal booked by each individual. There are three different statuses for each meal: Booked ('B'), Un-booked ('U'), and Taken ('T').

The "bookedmeal" table is the most frequently used table of the *"Mobile Meal Booking System"*. Most of the methods act upon this table, such as "bookMeal()", "unbookMeal()", "changeDiet()", "viewBook()" and so on.

Bookedmeal					
Field	**Type**	**Null**	**Key**	**Default**	**Extra**
bookedmealID	varchar(100)	NO	YES	NULL	
name	varchar(8)	NO		NULL	
Mealdate	Date	NO		NULL	
breakfast	Char(1)	YES		'N'	
statusB	Char(1)	YES		'U'	
lunch	Char(1)	YES		'N'	
statusL	Char(1)	YES		'U'	
supper	Char(1)	YES		'N'	
statusS	Char(1)	YES		'U'	

TABLE 3.5.2-3 – BOOKEDMEAL TABLE

3.6 Chapter review

In this chapter we took a look at the *"Rhodes University Mobile Meal Booking System"*, discussing the development and the deployment of the J2ME mobile Web Application in depth. First we looked at the hardware and software development environment. We then introduced the development of the system in three separate sections, which included front tier MIDlet development, middle tier Servlet development and backend database design. Each section covered a complete description of the development life cycle, from designing to coding, from testing to deployment.

Chapter 4: Developing .NET Mobile Web Applications

–*" Nothing is built on stone; all is built on sand, but we must build as if the sand were stone"* [Quotable Quotes].

This chapter covers an investigation into the development life cycle of .NET mobile Web applications and the case study of the *"Rhodes University Mobile Geographic Information System"*. This application provides detailed geographic information of the Grahamstown campus of Rhodes University via mobile devices such as cell phones and Pocket PCs. It presents the information visually, a campus map for instance. At the same time users can easily navigate through the entire application by using various navigational mechanisms such as navigation buttons, multilevel map buttons, location jump textboxes and so on [Source code, related material and video clip demos can be found on the accompanying CD-ROM].

4.1 Development platform and environment – Hardware

As mentioned in the previous chapter, both of the .NET and J2ME mobile applications created for this project share the same hardware platform and network environment [For a detailed introduction please refer to section 3.1].

4.2 Software environment installation and configuration

All software used to build the development platform for the "*Rhodes University Mobile Geographic Information System*" is proprietary software.

4.2.1 Software packages

4.2.1.1 Operating system:

Microsoft Windows XP Professional.

4.2.1.2 Application runtime environment:

Microsoft .NET framework (English) Version: 1.0.3705 (1800 Mbytes)

The Microsoft .NET Framework is the Microsoft Windows component for building and running various applications and XML Web services.

4.2.1.3 Web server:

Microsoft Internet Information Services (IIS) Version: 5.1 (15.7 Mbytes)

IIS is an integrated Internet service system that contains an HTTP server, an FTP server, an SMTP server and so on. It hosts and manages Web sites and shares information across the Internet.

4.2.1.4 Database server:

Microsoft SQL Server 2000 (evaluation) Version: 8.00.194 (91.48 Mbytes)

Microsoft SQL Server 2000 is a relational database management system. It also includes support for XML and HTTP.

4.2.1.5 IDE and toolkits:

1) Microsoft Visual Studio .NET Enterprise Architect (English), (1760 Mbytes)

Visual Studio .NET is a tool set for building and integrating XML Web services, Microsoft Windows–based applications, and Web solutions.

2) Microsoft Mobile Internet Toolkit (MMIT) Version 1.0.2506, (8.21 Mbytes)

The Microsoft Mobile Internet Toolkit is an extension of the .NET framework and Visual Studio .NET. It provides a visualized design environment for the development of mobile Web applications. It can be downloaded for free from the Microsoft Web Site.

3) Microsoft Mobile Explorer. Version: 3.00.4217, (4.49 Mbytes)

Microsoft Mobile Explorer is a mobile device emulator that allows developers to test their mobile Web applications.

4) Openwave SDK. Version: 5.1.

Openwave SDK is a set of development toolkits for WML enabled mobile devices. It also provides a range of mobile device emulators such as Siemens s45 and Open wave mobile browser 6.0.

4.2.2 Installation & configuration

All the development tools and servers provide "wizard" installation. Simply follow the instructions to complete installation. No manual configuration is required.

4.2.3 Investigation & optimisation

The installation process is time-consuming. It took approximately 4 hours and 3000 Mbytes disk space (excluding the Operating System) to complete the development environment. Following the Microsoft price list[12], it costs "US$ 21,377" (including the Operating System) to build the entire development platform. The hardware configurations required reach the minimum requirement as the most resource and computation intensive application – Visual Studio.NET.

4.3 The *"Mobile Geographic Information System"*

The following sections introduce a range of .NET technologies used to build mobile Web applications. They also observe the development life cycle in detail of the "*Rhodes University Mobile Geographic Information System*", a .NET mobile Web application created for this project.

4.3.1 "RU Mobile GIS" application overview

The *"Rhodes University Mobile Geographic Information System"* is a .NET Mobile Web Application. It delivers geographic information on Rhodes University campus to various mobile devices. This system is a three-tier system and it contains the following layers:

4.3.1.1 Application architecture overview

Client tier

The client tier is made up of mobile Web browsers. The *"RU Mobile GIS"* is designed to run on a wide range of different mobile devices, and the diversity of mobile Web browsers is considered. *"RU Mobile GIS"* makes no effort to control or modify the client browser behaviour. In fact, only markups are transferred to the client-side and displayed according to the browser.

Middle tier

The Middle tier is the main part of the *"RU Mobile GIS"*. It resides on the Web server and contains methods of application logic. It includes three classes: the "GISLogin" module, the "GISServer" module and the "WMLGISServer" module.

[12] Windows XP Professional US$ 299; Visual Studio.NET Enterprise Edition US$ 1,079 and SQL Server Enterprise Edition US$ 19,999 [Microsoft].

Backend tier

The backend tier maintains the connectivity to the database, executes queries based on the middle tier requests and returns the results accordingly. The backend tier has two components. They are "AccountDataComp" and "MapDataComp".

4.3.1.2 Language choice

The code in "RU Mobile GIS" is written in C# (C sharp). C# is a new language created by the Microsoft development team. It is a powerful and flexible programming language that can be used to create a variety of applications, including dynamic Web applications. C# is a fully object-oriented programming (OOP) language. The syntax of C# is similar to that of C++ and Java.

4.3.1.3 Application class hierarchy overview

Figure 4.3.1-1 shows the hierarchical relationship of all the classes that have been developed for the *"RU Mobile GIS"* within the .NET framework.

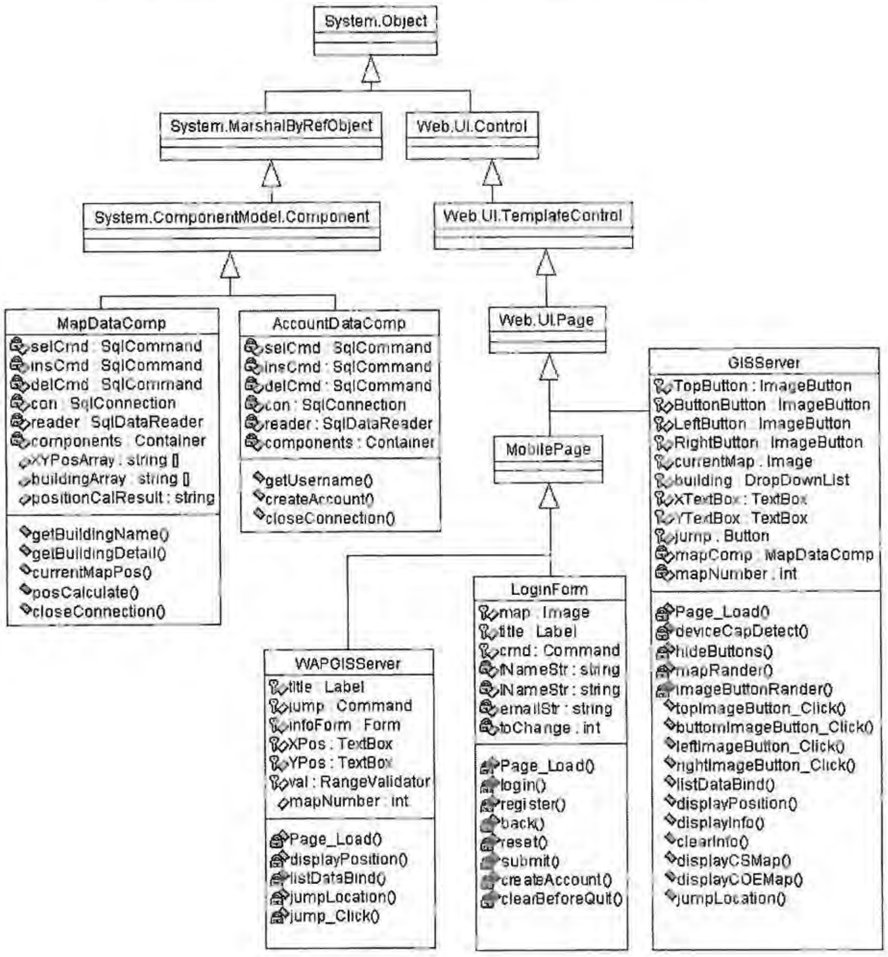


FIGURE 4.3.1-1 – *"RU MOBILE GIS"* CLASS HIERARCHY

4.4 Developing the Web component

4.4.1 Mobile ASP.NET

ASP.NET is a programming framework, built on the common language runtime (CLR) that can be used on a server to build Web applications. ASP.NET is not just a simple extension of ASP; it allows developers to use a full featured programming language, such as C# or VB.NET, to build Web applications in an object oriented fashion. An ASP.NET application is typically made up of Web pages that have the extension ".aspx", for example the "GISServer.aspx", and code modules that handle events, provide business logic and perform some other functions as well.

4.4.1.1 ASP.NET mobile Web control model[13]

Web Controls are the most frequently used objects for building .NET mobile Web applications. They are used to build user interfaces and to associate user interaction with application logic.

1) Device capabilities

a) Traditional Web applications

Most traditional Web applications are designed specifically for PCs on a limited choice of browsers such as Internet Explorer or Netscape. These browsers take the full power from the PCs and use a unified language – HTML. Therefore, most development for traditional Web applications use HTML Web controls and focus on the presentation of the application.

b) Mobile Web applications

Mobile devices use a wide array of browsers. Each of these browsers has distinct characteristics, such as screen size, screen resolution and colour capability. Furthermore, these browsers use a variety of very different markup languages, which include HTML, WML and cHTML. Each of these markup languages has a different set of controls.

2) Mobile Web control abstraction

ASP.NET uses the concept of abstraction in the development of mobile Web applications. Abstract Web controls are objects that represent the fundamental conceptual components of a visual display and functionality, such as command

[13] A Web control is an interface that allows users to interact with a Web application, a button for example.

buttons and text fields. When developing mobile Web applications the abstract controls are presented using XML [See Code 4], which cannot be used directly on any of the mobile browsers. At runtime the abstract representations (XML) are translated into device-specific Web controls using the corresponding markup language (WML, HTML).

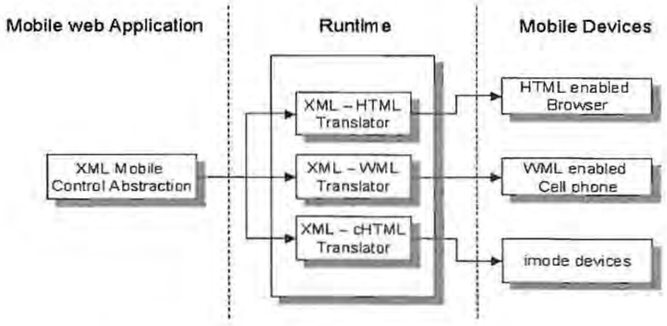


FIGURE 4.4.1-1 – MOBILE CONTROL ABSTRACTION

3) ASP.NET mobile Web controls

Mobile ASP.NET provides a rich collection of mobile controls for construction of mobile Web applications. Figure 4.4.1-2 shows a hierarchical view of ASP.NET mobile controls.

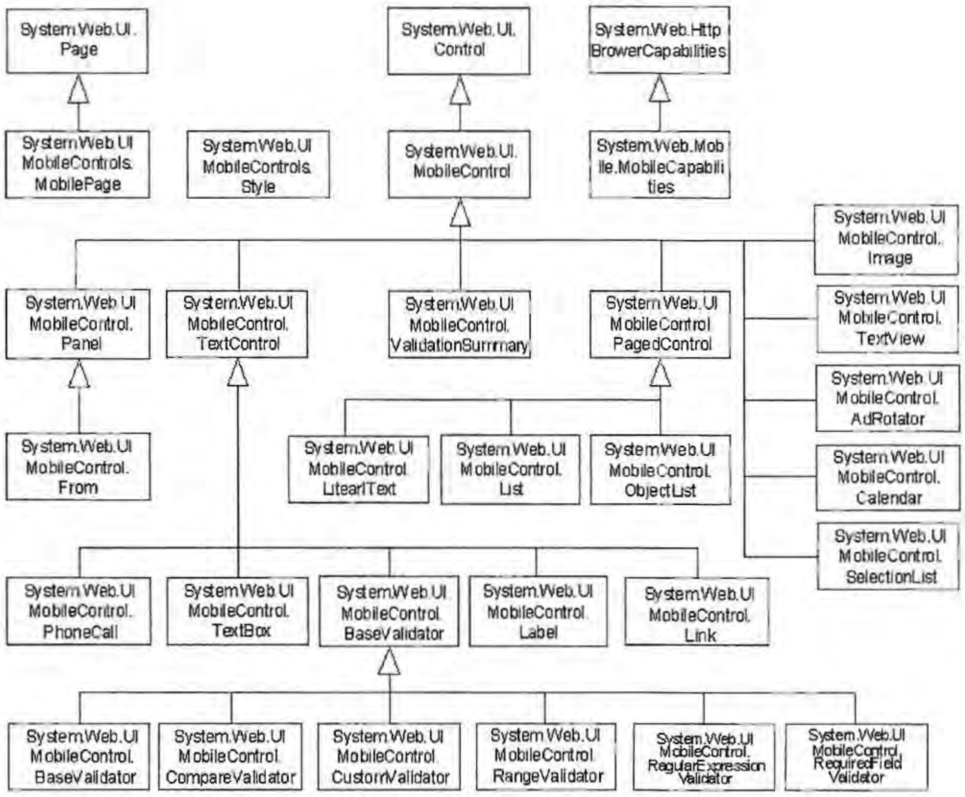

System.Web.UI. Page
System.Web.UI. Control
System.Web.Http BrowerCapabilities
System.Web.UI MobileControls. MobilePage
System.Web.UI MobileControls. Style
System.Web.UI. MobileControl
System.Web.Mobile.MobileCapabilities
System.Web.UI MobileControl. Image
System.Web.UI MobileControl. Panel
System.Web.UI MobileControl. TextControl
System.Web.UI MobileControl. ValidationSummary
System.Web.UI MobileControl PagedControl
System.Web.UI MobileControl. TextView
System.Web.UI MobileControl. AdRotator
System.Web.UI MobileControl. From
System.Web.UI MobileControl LitearlText
System.Web.UI MobileControl List
System.Web.UI MobileControl ObjectList
System.Web.UI MobileControl. Calendar
System.Web.UI MobileControl SelectionList
System.Web.UI MobileControl. PhoneCall
System.Web.UI MobileControl. TextBox
System.Web.UI MobileControl BaseValidator
System.Web.UI MobileControl Label
System.Web.UI MobileControl. Link
System.Web.UI MobileControl. BaseValidator
System.Web.UI MobileControl CompareValidator
System.Web.UI MobileControl CustomValidator
System.Web.UI MobileControl RangeValidator
System.Web.UI MobileControl. RegularExpression Validator
System.Web.UI MobileControl. RequiredField Validator

FIGURE 4.4.1-2 – ASP.NET MOBILE CONTROLS

ASP.NET Mobile controls can be classified into the following groups by categorizing their functionalities.

a) Fundamental controls

Fundamental controls are mobile versions of standard ASP.NET controls. They are “Form”, “Command”, “Label”, “TextBox”, “TextField” and “Image”.

b) Mobile specific controls

Mobile specific controls are those that apply only to mobile Web applications, for which there are no corresponding ASP.NET controls. For example, the “PhoneCall” is a Mobile specific control.

c) Validation controls

Validation controls are those used to validate user input. For example, the “RequiredFieldValidator” is used to check if the input area is empty.

d) Style controls

Style controls define the appearance of those controls to which it applies, for example font or colour.

e) DeviceSpecific controls

DeviceSpecific controls give developers the ability to customize controls for specific devices.

f) Template controls

With template controls default output can be overridden to render additional information.

4) Container: Form & Panel

For any .NET mobile Web application the root element is a mobile page, which is represented as the MobilePage object. For the *“RU Mobile GIS”* the MobilePage objects are “GISServer.aspx”, “GISCell.aspx”, and “GISLogin.aspx”. Within each MobilePage there are one or more “Form” controls. A Form is the outermost container that groups other controls logically, all other controls must be placed inside a Form control. Form controls cannot be nested. Another type of container is a “Panel”, which can be placed inside a Form control. A Form control can contain one or more Panel controls. Unlike Form controls, Panel controls can be nested.

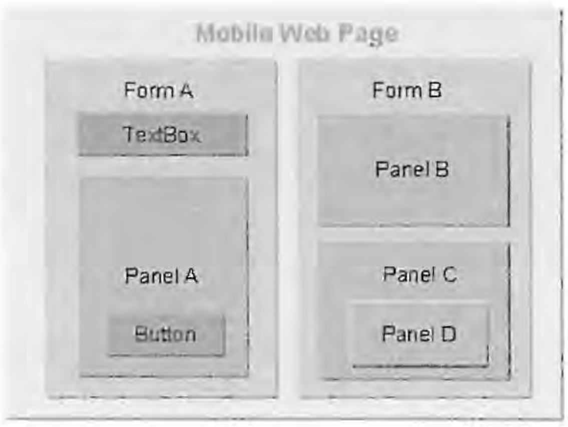


FIGURE 4.4.1-3 – MOBILE ASP.NET CONTAINERS

5) Mobile ASP.NET event handling model

The majority of mobile Web applications are event-driven. When a user interacts with a mobile control an event is triggered and a specific piece of code executeed to respond to the event and perform the action.

a) Server-side event handling

Unlike other common applications, events generated from .NET mobile Web applications are handled only on the server side. That is to say that each time the user performs an action such as clicking a button or entering some text into a text box, events are triggered. On the client side these events cause no action other than an HTTP POST request, containing details of these events, to the mobile Web application server. The server receives the request and executes the server-side code to perform the requested action accordingly. The server then sends the client a response containing updated information. Finally the client refreshes its display to complete the transaction. Figure 4.4.1-4 demonstrates the event-handling process.

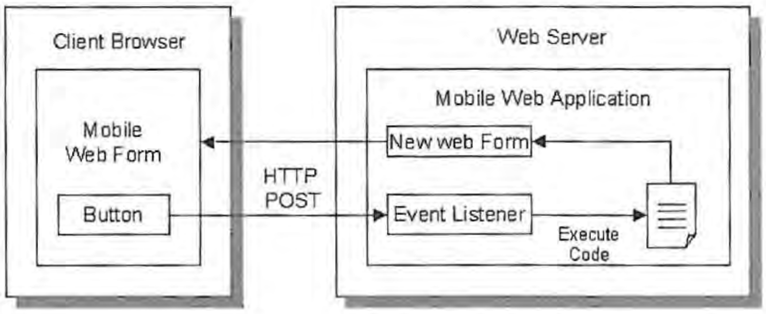


FIGURE 4.4.1-4 – MOBILE ASP.NET EVENT HANDLING MODEL

b) Hidden event delegates

Usually, when programming with events, event delegates are used to set up the connection between the event generators and the event handlers. To handle a specific event, a corresponding delegate is registered with the event generator. The event can thereby be redirected to the event handler. In *"RU Mobile GIS"*, however, event handler methods are written without too much concern for how the actual events and their respective handler methods connect. The connection details are hidden and separated by the .NET framework and the Visual Studio.NET. There are two ways to do this.

i) Mobile ASP.NET runtime

By assigning the "AutoEventWireup" attribute of the "@Page" directive to an ASP.NET mobile Web page, the mobile ASP.NET automatically wires up the events and event handlers at runtime.

ii) Visual Studio.NET

Alternatively, by using Visual Studio .NET, developers need not write specific code for wiring events and event handlers. The IDE will insert the necessary code automatically.

4.4.1.2 Mobile ASP .NET session management

As aforementioned, when building dynamic mobile Web applications, information needs to be stored between client requests and server responses. The underlying layer of Web transmission protocol – HTTP - is stateless and thus does not support inherited states management. Therefore alternative mechanisms are used.

Besides the support of the traditional approaches introduced in section 3.4.3 of the previous chapter, .NET offers a range of other distinct mechanisms for preserving and maintaining state information.

1) Session state

The "Session state" maintains the variables and objects for a client over multiple requests and responses.

2) Application state

The "Application state" preserves the variables and objects of an application over multiple requests by multiple clients.

3) View state

View state stores the values of a mobile Web Forms page on the server.

In the *"RU Mobile GIS"*, "ViewState" is used for passing values between the client browser and the GISServer. Code 4.4.1-1 shows a example of using "ViewState" to store the value of a variable.

```
...
public int mapNumber{
      get{ return (int)ViewState["mapNumber"];}
      set{ ViewState["mapNumber"] = value;}
}...
```

CODE 4.4.1-1 – USING VIEWSTATE

4.4.2 .NET mobile Web application development life cycle

The introduction into the development life cycle of the "RU Mobile GIS" still follows the same steps of:

designing → programming → testing → deploying.

4.4.2.1 STAGE ONE: Designing the Mobile Web Page

Step 1: Define design goals:

The list of goals to be achieved as a completed system:

1) To develop a fully functional Mobile Geographic Information System.

2) Geographic information should be presented to users in multiple ways, such as in graphical representation (map) and literal presentation (Text).

3) Geographic Information should be presented in a meaningful and sententious manner.

4) User interfaces should be straightforward and easy to understand.

5) The application should be flexible enough to run on a wide range of mobile devices, which have different capabilities.

Step 2: Categorize functional modules:

The *"RU Mobile GIS"* is a typical event-driven application. Therefore most of its functions are triggered by user actions. Generally there are three major types of functions: navigation, information fetching and display rendering.

1) Navigation

Then Navigation function can be further categorized into the following modules:

a) Button navigation

In a Geographic Information System a map is the best choice for presenting visual information to the user. Buttons can be used as well to navigate through the map.

b) Parameter navigation

The physical location of a Geographic Information System is represented using coordinates and parameters. Therefore, the system should provide users a way to navigate by entering coordinate parameters.

c) GPS navigation

A true Geographic Information System usually takes coordinate parameters from a Global Positioning System (GPS) and provides real time information. Unfortunately, due to a lack of hardware facilities, this approach cannot be adopted in the *"RU Mobile GIS"*. An interface is provided for such a future extension.

2) Information fetching

The primary goal of a Geographic Information System is to present location information to the user. The information-fetching module retrieves the corresponding information from the backend database according to the request.

3) Display rendering

After the user's location has been identified the screen should render the relevant image. The display-rendering module takes on this responsibility.

Step 3: Abstract interactions between functional modules:

Figure 4.4.2-1 illustrates the interactions and relationships between modules by the use of a flow chart.

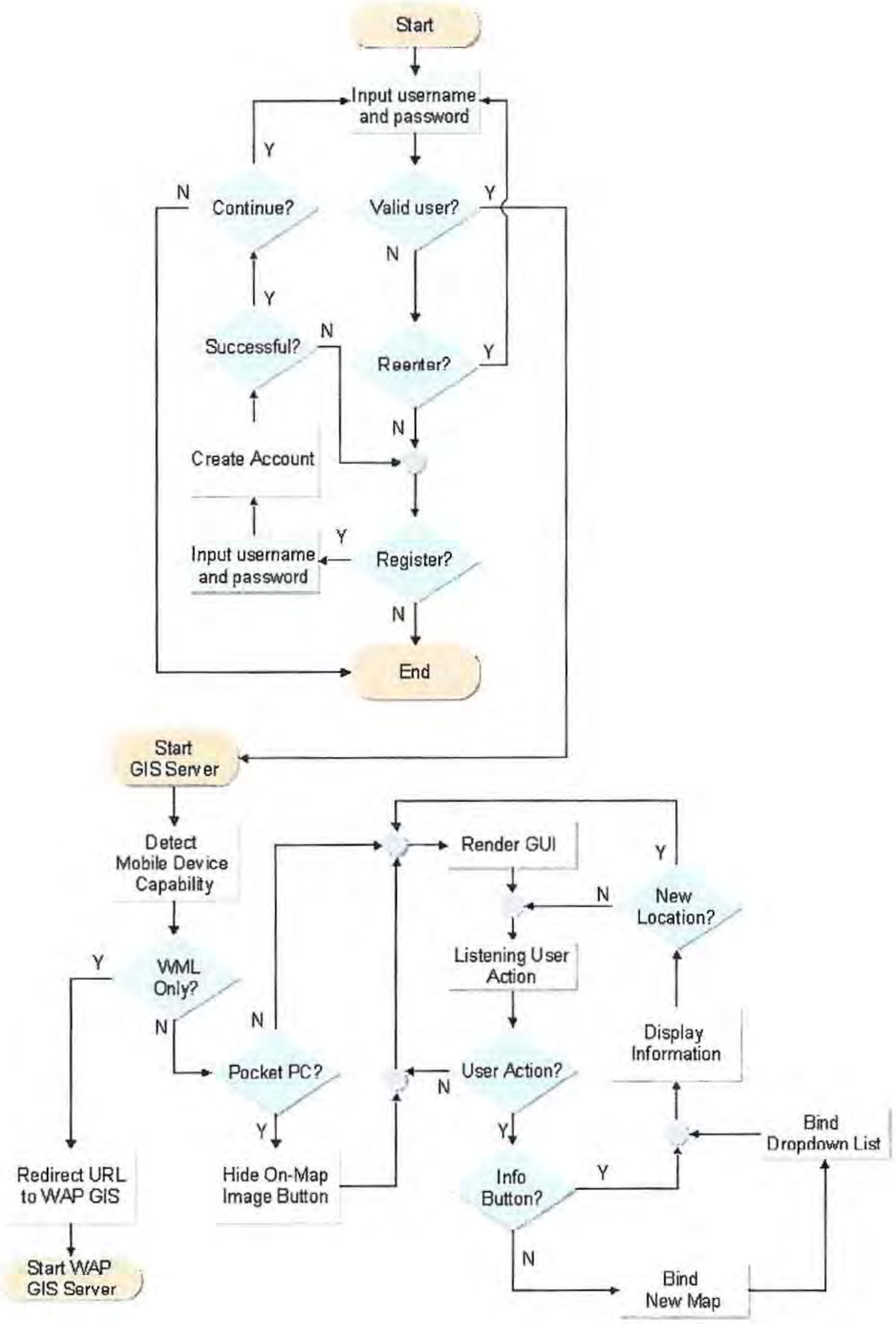


FIGURE 4.4.2-1 – "RU MOBILE GIS LOGIC FLOW CHART

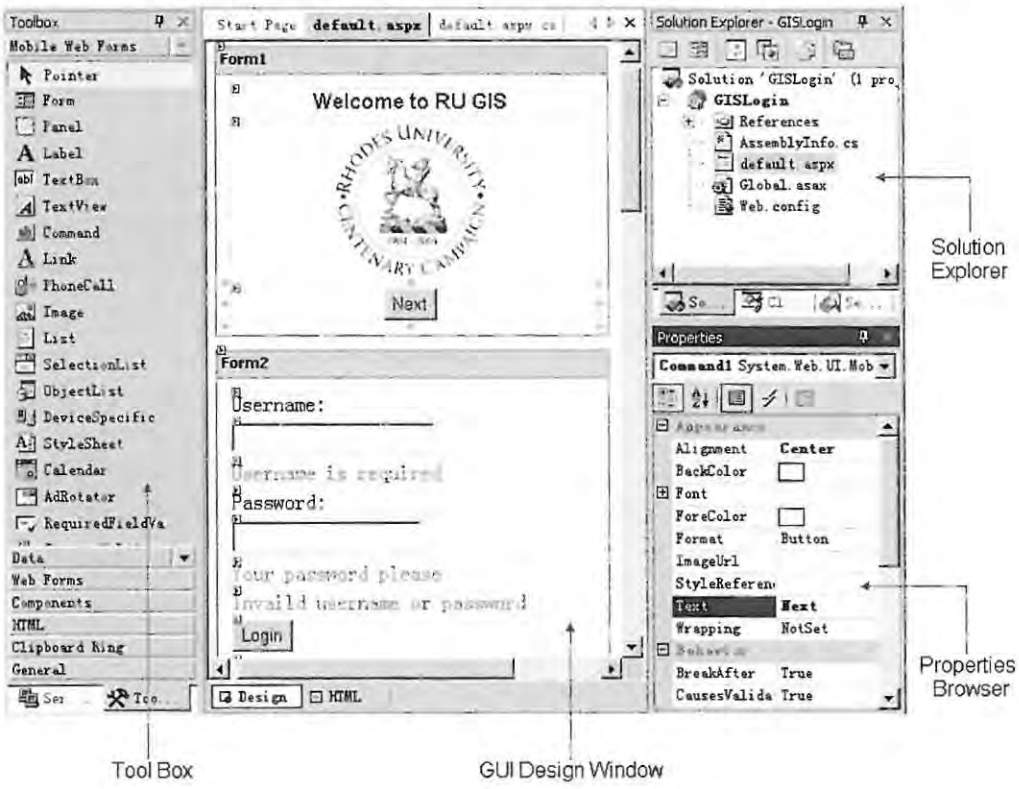
Step 4: Design user interfaces and GUI layout:

1) Positioning and layout of Mobile Form

As already stated, mobile devices are limited by their display capabilities. Unlike traditional Web applications that target PCs, in designing mobile Web applications there is no such concept as a grid-layout. Controls in a mobile form can only be placed in a sequential manner, from the top down. Controls are placed in a mobile form rather for their logical functionalities than for aesthetic purposes [Please refer to section 4.4.1.1 for the concept of the control abstraction]. Therefore a concern with details of physical appearance such as changing vertical spacing between controls or resizing a control is inappropriate here.

2) Using Visual Studio.NET to design GUI components

As shown in figure 4.4.2-2, Visual Studio.NET provides a graphically integrated development environment for designing GUI components for mobile Web applications. The graphical representation of GUI components obeys the positioning rules of a .NET mobile Web Form. In this way it cannot be classified as a WYSIWYG[14] development environment.



Command1 System.Web.UI.Mob

```
Appearance
  Alignment        Center
  BackColor        □
⊞ Font
  ForeColor        □
  Format           Button
  ImageUrl
  StyleReferen
  Text             Next
  Wrapping         NotSet
Behavior
  BreakAfter       True
  CausesValida     True
```

FIGURE 4.4.2-2 – VS.NET MOBILE WEB APPLICATION DEVELOPMENT ENVIRONMENT

[14] WYSIWYG is the acronym for What You See Is What You Get

a) Tool box

Tool box provides sets of controls that can be dragged and dropped into the GUI Design Window to construct a mobile Web application.

b) GUI design window

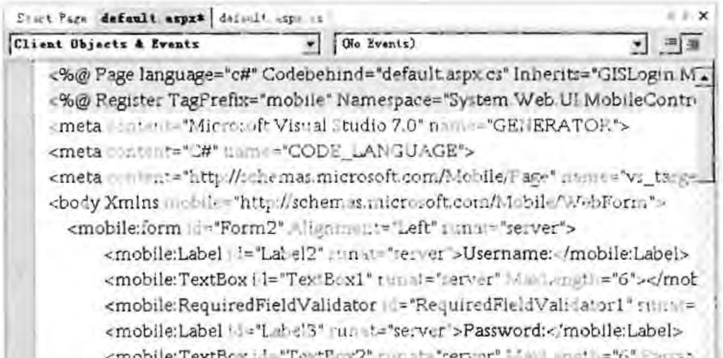
The GUI design window is the main work area for designing the graphical user interfaces of an application. By using GUI design window, controls can be logically grouped together and as a result display sequence and event actions can be assigned. Clicking the "HTML"[15] tag on the bottom left-hand corner of the GUI design window brings up the source code of the current mobile Web page [Please see figure 4.4.2-3].

c) Solution explorer

Solution explorer contains a list of files that are used within the current application, including "mobile Web page", "code behind", "configuration file" and so forth. Double clicking the nametag of a file brings the contents of the file into the design window.

d) Properties browser

When clicking on a control in the GUI design window, the properties (such as Text, URL or Font) of this control display inside the properties browser. To change these properties, one simply modifies the value in the text box next to the property name.



```
<%@ Page language="c#" Codebehind="default.aspx.cs" Inherits="GISLogin.M
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileContr
<meta content="Microsoft Visual Studio 7.0" name="GENERATOR">
<meta content="C#" name="CODE_LANGUAGE">
<meta content="http://schemas.microsoft.com/Mobile/Page" name="vs_targ
<body Xmlns mobile="http://schemas.microsoft.com/Mobile/WebForm">
  <mobile:form id="Form2" Alignment="Left" runat="server">
        <mobile:Label id="Label2" runat="server">Username:</mobile:Label>
        <mobile:TextBox id="TextBox1" runat="server" MaxLength="6"></mot
        <mobile:RequiredFieldValidator id="RequiredFieldValidator1" runat=
        <mobile:Label id="Label3" runat="server">Password:</mobile:Label>
        <mobile:TextBox id="TextBox2" runat="server" MaxLength="6" Passw
        <mobile:RequiredFieldValidator id="RequiredFieldValidator2" runat=
```

FIGURE 4.4.2-3 – MOBILE WEB PAGE SOURCE CODE VIEW

[15] More accurately, it should be termed, "Source Code" instead of "HTML". Source code of a .NET mobile Web page is written in XML as we mentioned in section 4.4.1.1.

4.4.2.2 STAGE TWO: Programming the "RU Mobile GIS"

Step 1: Coding

"RU Mobile GIS" is an event-driven application. Methods are invoked when events are triggered and handled. Besides all the event handling methods, such as "Command1_Click()" and "Page_Load()" mentioned in section 4.4.2.1, there are three types of functional methods. These are: Navigation methods, Information fetching methods and Display rendering methods.

The *"RU Mobile GIS"* is made up of several sub systems, which include "Authentication & Authorization Module", "GIS Information Provider Module", "Data Supplier Module" (to be discussed in section 4.5) and "Distribution & Installation Module" (to be introduced in section 4.4.2.4). "GIS Information Provider Module" contains two units, one being the "GISServer" that provides full functionality of the system with various ways of presenting information for high-end mobile devices (such as Pocket PCs). The other is the "WMLGISServer", which is a reduced subset of the "GISServer" that offers limited functionalities and literal only information presentation. The following introduction concentrates on the "GISServer".

Coding the "GISServer"

1) Coding event handler methods

As pointed out in section 4.4.1.1, the .NET Framework hides the implementation details of registration and the communication between events and event-delegates or event-handlers, from the developer. Hence, there is no need to worry about the event transmission pipeline. Better yet, event handling can be added to a mobile control by simply double clicking on the control itself in the GUI design window. The Visual Studio.NET will automatically switch to the source code window and insert the skeleton of the handler. Here is an example of the event handler method:

```
private void TopLeftImageButton_Click(object sender, ImageClickEventArgs e){
        switch(mapNumber){
            ...
        }
        ImageButtonRander();
        ListDataBind();
        displayPosition();
}...
```

CODE 4.4.2-1 – EVENT HANDLER METHOD

2) Coding functional methods

a) Navigation methods

i) "GoTopLeft()"

Methods like "GoTopLeft()" provide on-screen navigation based on user actions.

ii) "JumpLocation()"

"JumpLocation()" takes coordinate parameters from on-screen text boxes and changes the location accordingly.

b) Information fetching methods

i) "ListDataBind()"

Methods like "ListDataBind()" call the backend data provider, "MapDataComp", and fetch information from the database according to the location.

ii) "DeviceCapDetect()"

"DeviceCapDetect()" method uses the "MobileCapabilities" object offered by .NET Mobile to gather information from the requesting mobile devices and thereby determine what type of content should be sent back, based on device capabilities.

```
...
private void DeviceCapDetect
{
  MobileCapabilities capabilities;

  capabilities = (MobileCapabilities)Request.Browser;

  string strBrowser = capabilities.Browser.ToString();
  string strIsColor = capabilities.IsColor.ToString();
  string strScreenWidth = capabilities.ScreenPixelsWidth.ToString();
  string strScreenHeight=capabilities.ScreenPixelsHeight.ToString();
     ...
}...
```

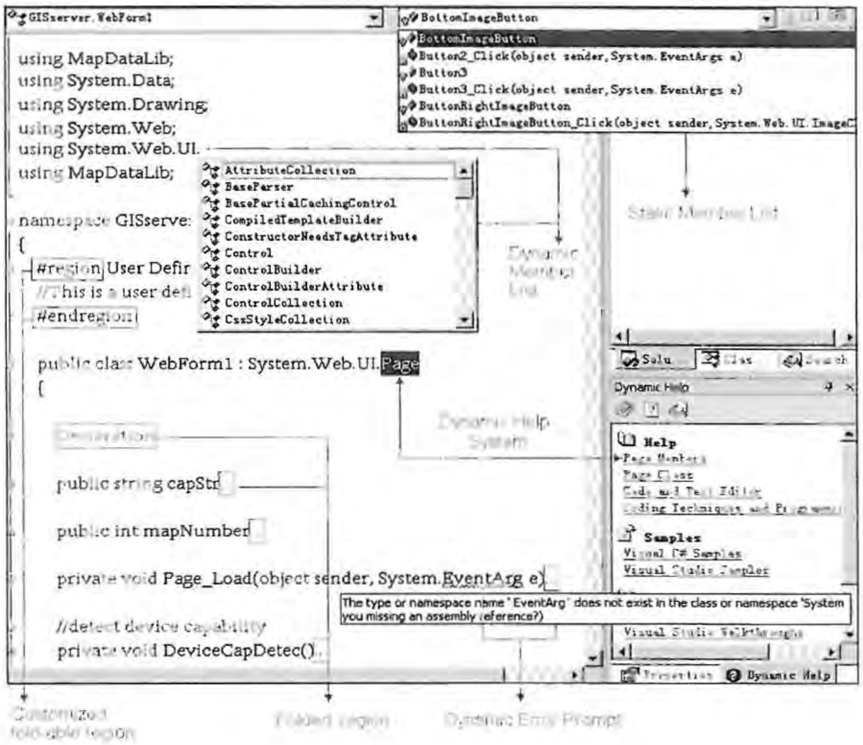
CODE 4.4.2-2 – DEVICECAPDETECT METHOD

c) Display rendering methods

Display rendering methods such as "ImageRandering()" and "ImageButtonRander()" display the GUI components to the user, hide those that are not supported by the requesting device and re-render the screen display each time the location changes.

Working with Visual Studio.NET

As shown in figure 4.4.2-4, Visual Studio.NET gives the developer a neat, comfortable, customisable and productive coding environment.



```
using MapDataLib;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.UI.
using MapDataLib;

namespace GISserve
{
  #region User Defir
    //This is a user defi
  #endregion

    public class WebForm1 : System.Web.UI.Page
    {

        Declarations

        public string capStr

        public int mapNumber

        private void Page_Load(object sender, System.EventArg e)

        //detect device capability
        private void DeviceCapDetec()
```

```
GISserver.WebForm1
AttributeCollection
BaseParser
BasePartialCachingControl
CompiledTemplateBuilder
ConstructorNeedsTagAttribute
Control
ControlBuilder
ControlBuilderAttribute
ControlCollection
CssStyleCollection
```

```
BottonImageButton
BottonImageButton
Button2_Click(object sender,System.EventArgs e)
Button3
Button3_Click(object sender,System.EventArgs e)
ButtonRightImageButton
ButtonRightImageButton_Click(object sender,System.Web.UI.ImageC
```

```
The type or namespace name 'EventArg' does not exist in the class or namespace 'System
you missing an assembly reference?)
```

FIGURE 4.4.2-4 – VISUAL STUDIO.NET CODING ENVIRONMENT

In addition to those traditional IDE tools such as Dynamic/Static Member Lists and Dynamic Error Prompt, Visual Studio.NET introduces some new features to the coding environment.

1) Fold-able region

Classes, methods, comments and even customized sections in the Visual Studio.NET code editor are encapsulated into many folder-like regions. These regions can be folded or unfolded as needed. As a result, programmers no longer need to scroll through thousands of lines to read the code. Obviously this improves code clarity and legibility significantly.

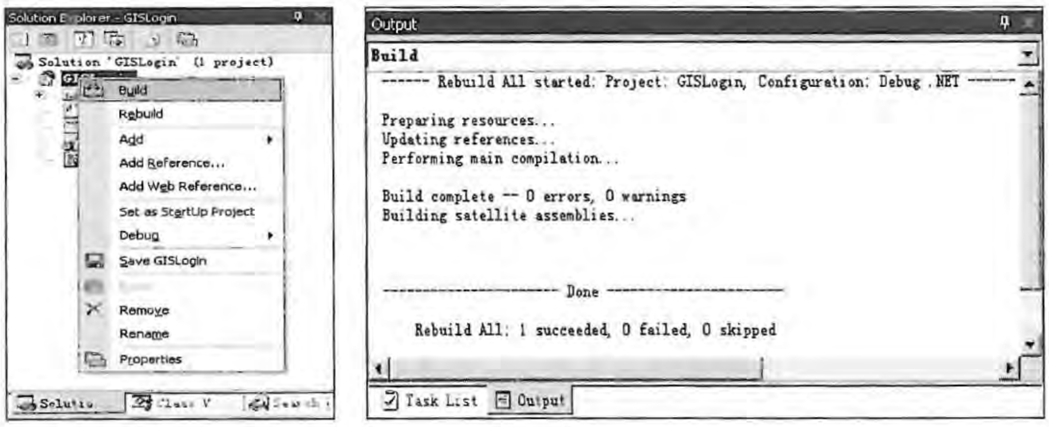
2) Dynamic help system

When highlighting a key word in the coding window, a list of information that relates to the key word, such as API, coding technique and samples, is displayed in the dynamic help window. Clicking on the link of the related topic causes the relevant information to be displayed inside the main windows as a new tag. This saves programmers from searching through the help system for help topics.

Step 2: Compile the application

There are two ways to compile a .NET mobile Web application: either by using a command prompt based compiler or by using the graphical utility in Visual Studio.NET. The second approach is chosen for compiling the *"RU Mobile GIS"*.

As shown in figure 4.4.2-5, compiling is begun by selecting the current project from the Solution Explorer. Right click to bring out the pop-up menu and select the Build option. The compiling status will be reflected in the Output window.



```
------ Rebuild All started: Project: GISLogin, Configuration: Debug .NET ------

Preparing resources...
Updating references...
Performing main compilation...

Build complete -- 0 errors, 0 warnings
Building satellite assemblies...



---------------------- Done ----------------------

    Rebuild All: 1 succeeded, 0 failed, 0 skipped
```

FIGURE 4.4.2-5 – COMPILING THE APPLICATION

4.4.2.3 STAGE THREE: Testing the "RU Mobile GIS"

The *"RU Mobile GIS"* is tested on both mobile device emulators and on real devices. The Openwave emulator is chosen for simulating the WML enable cell phone while the Compaq Pocket PC is also used to test this application.

Testing with mobile devices

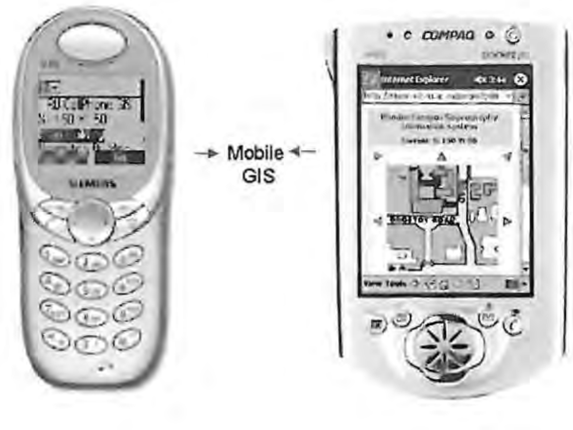
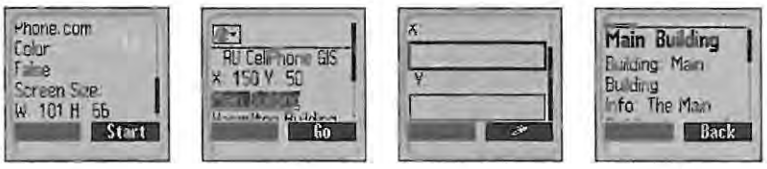


FIGURE 4.4.2-6 – TESTING THE "RU MOBILE GIS" ON MOBILE DEVICES

1) Testing with Siemens s45 WML cell phone emulator

As shown in figure 4.4.2.7, Siemens s45 has a monochrome screen and the Web browser running on it supports only WML markup and ".wbmp" image format.



a *b* *c* *d*

FIGURE 4.4.2-7 – TESTING THE "RU MOBILE GIS" ON SIEMENS S45

a) Screen *a* displays the mobile device information such as the Web browser name (Phone.com), colour support (false), screen size (101 pixel × 66 pixel) and so forth.

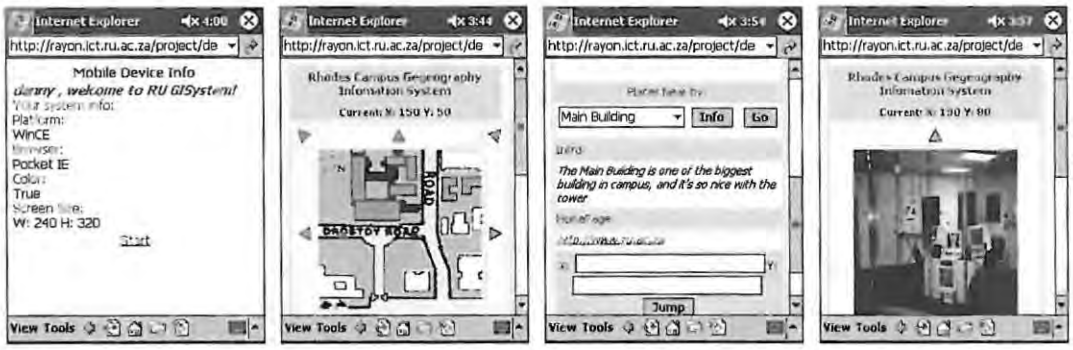
b) Screen *b* is the main screen of the *"RU Mobile GIS"* WML version. It displays the current coordinate, (X, Y), a list of buildings within range of the location, and a button that is linked to screen *c*.

c) Screen *c* shows the navigator which takes X and Y as parameters and then modifies the location accordingly.

d) Screen *d* is a textual description of the selected building.

2) Testing with Compaq Pocket PC:

The Web browser running on Pocket PC is Microsoft Pocket Internet Explorer, an HTML browser that is supported by Mobile .NET. Compared with the Siemens s45 cell phone, testing the *"RU Mobile GIS"* with Pocket PC is much more interesting. Pocket PC has a bigger screen (240 pixel×320 pixel), supports colour display and multiple image format that allows a more flexible content to be displayed. The basic functionality is much the same, but when running "RU Mobile GIS" on Pocket PC, the map (interactive, three levels in depth), navigation buttons, and URL links are provided.



danny , welcome to RU GISystem!
Your system info:
Platform:
WinCE
Browser:
Pocket IE
Color:
True
Screen Size:
W: 240 H: 320

The Main Building is one of the biggest building in campus, and it's so nice with the tower

FIGURE 4.4.2-8 – TESTING THE "RU MOBILE GIS" ON POCKET PC

Using trace facility

The Mobile ASP.NET gives developers a Web based application testing and debugging tool – the Trace facility. Using the Trace facility, debugging and testing statements can be inserted into the code to print out information such HTTP connection status, transmission content, session states, session variables, application execution times, and the entire control object hierarchy.

The tracing can be enabled at two different levels, the page-level and the application-level.

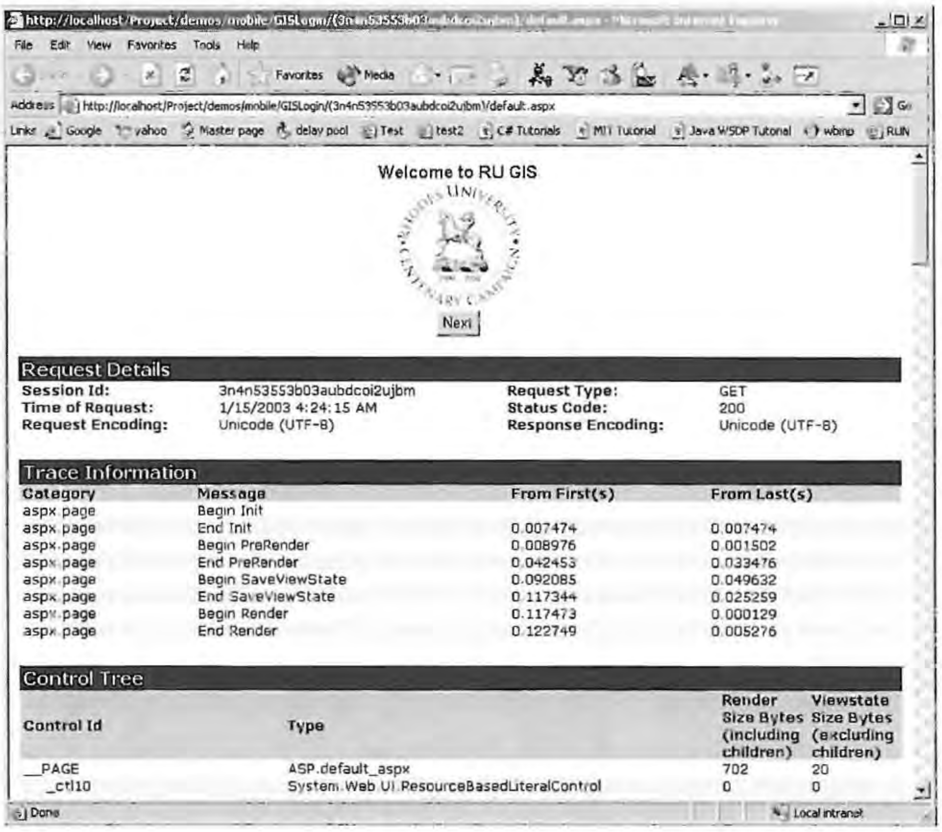
1) Page-level tracing

To enable page-level tracing, set the "Trace" attribute to true within the "@Page" directive at the top of the Mobile ASP.NET page.

```
<%@ Page Trace = "true" language="c#" Codebehind="default.aspx.cs"
Inherits="GISLogin.MobileWebForm1" AutoEventWireup="false" %>
```

CODE 4.4.2-3 – ENABLING PAGE LEVEL TRACING

In page-level tracing mode trace messages are appended to the current page output and displayed as an HTML page.



Request Details

Session Id:	3n4n53553b03aubdcoi2ujbm	**Request Type:**	GET
Time of Request:	1/15/2003 4:24:15 AM	**Status Code:**	200
Request Encoding:	Unicode (UTF-8)	**Response Encoding:**	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin Init		
aspx.page	End Init	0.007474	0.007474
aspx.page	Begin PreRender	0.008976	0.001502
aspx.page	End PreRender	0.042453	0.033476
aspx.page	Begin SaveViewState	0.092085	0.049632
aspx.page	End SaveViewState	0.117344	0.025259
aspx.page	Begin Render	0.117473	0.000129
aspx.page	End Render	0.122749	0.005276

Control Tree

Control Id	Type	Render Size Bytes (including children)	Viewstate Size Bytes (excluding children)
__PAGE	ASP.default_aspx	702	20
_ctl10	System.Web.UI.ResourceBasedLiteralControl	0	0

FIGURE 4.4.2-9 – PAGE LEVEL TRACING OUTPUT

2) Application-level trace

To enable application-level trace logging, edit the "web.config" file in the application root directory. Also set the "enabled" attribute of "trace" element to true.

```
<configuration>
  <system.web>
     <trace enabled="true" requestLimit="10" pageOutput="false"/>
  </system.web>
</configuration>
```

CODE 4.4.2-4 – ENABLING APPLICATION LEVEL TRACING

In application-level tracing mode trace messages are stored on an HTML log file that can be accessed through the URL: http://hostname(fully qualified URL of the destination system)/GISServer/ Trace.axd

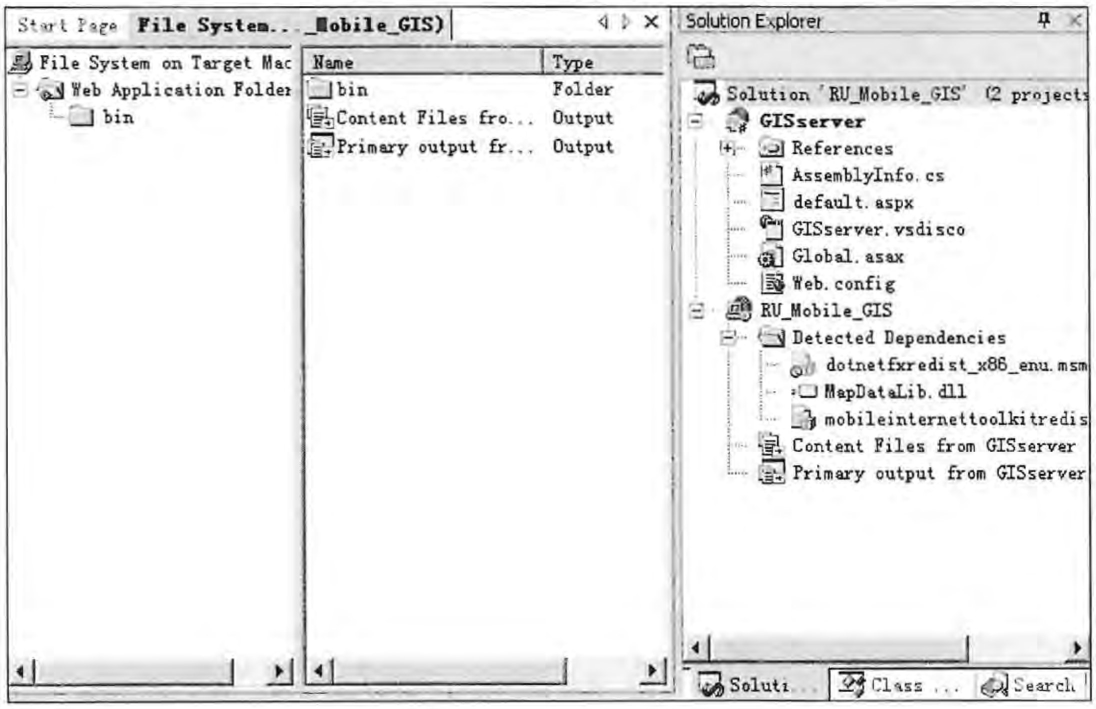
4.4.2.4 STAGE FOUR: Packing and deploying "RU Mobile GIS"

After successfully building the *"RU Mobile GIS"* it is the time to take a further step – packaging the application, and releasing it for distribution into a production system.

To deploy a simple application that has no dependence on other assemblies[16], simply copy it into the destination system. For a more complex application, such as the *"RU Mobile GIS"*, Visual Studio.NET offers facilities to create a windows installer that can be executed on the destination system.

1) Packaging the "RU Mobile GIS"

To create a windows installer a Web setup project must be created first. Creating a Web setup project using Visual Studio.NET is no different from building other .NET applications. As elsewhere, Visual Studio.NET provides a set of utilities with which to perform the necessary operations. Figure 4.4.2-10 shows the development environment for the "RU_Mobile_GIS" Web setup project.



```
File System on Target Mac
- Web Application Folder
    bin
```

Name	Type
bin	Folder
Content Files fro...	Output
Primary output fr...	Output

```
Solution 'RU_Mobile_GIS' (2 projects
- GISserver
    + References
      AssemblyInfo.cs
      default.aspx
      GISserver.vsdisco
      Global.asax
      Web.config
- RU_Mobile_GIS
    - Detected Dependencies
        dotnetfxredist_x86_enu.msm
        MapDataLib.dll
        mobileinternettoolkitredis
      Content Files from GISserver
      Primary output from GISserver
```

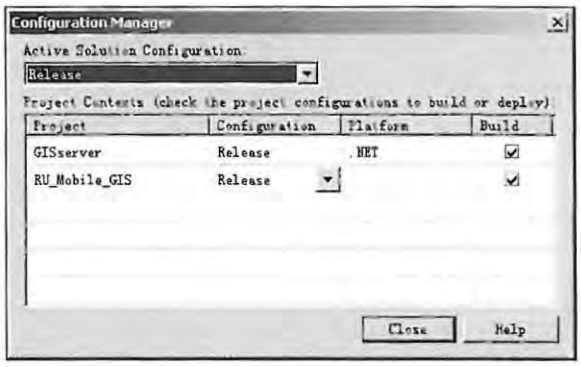
FIGURE 4.4.2-10 – WEB SETUP PROJECT

[16] Assemblies are modular components used to build applications. They are similar to traditional windows DLLs and java classes.

Procedure

a) Add the *"RU Mobile GIS"* projects into the Web setup project by selecting the "Add Existing Project" option from the "File" menu.

b) Using the Configuration Manager change the project configuration to "Release".



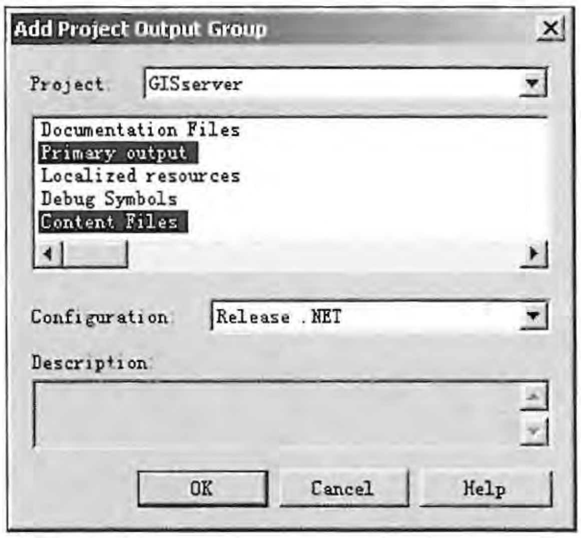
Active Solution Configuration: Release

Project Contents (check the project configurations to build or deploy):

Project	Configuration	Platform	Build
GISserver	Release	.NET	✓
RU_Mobile_GIS	Release		✓

FIGURE 4.4.2-11 – CONFIGURATION MANAGER

c) Add the project primary output and content files from the "GISServer" project to the "RU_Mobile_GIS" Web setup project.



Project: GISserver

Documentation Files
Primary output
Localized resources
Debug Symbols
Content Files

Configuration: Release .NET

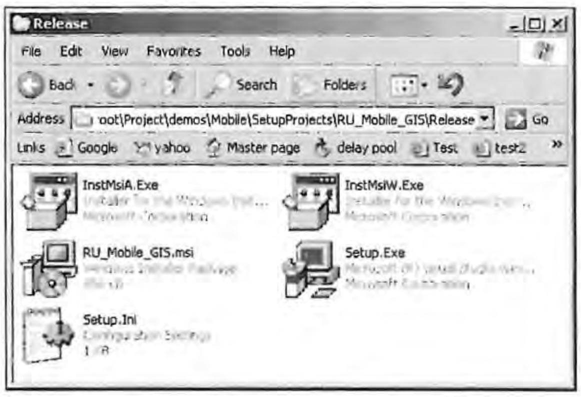
Description:

FIGURE 4.4.2-12 – ADD PROJECT OUTPUT

d) Finally compile and build the "RU_Mobile_GIS" Web setup project and generate the windows installer.

2) Deploying the "RU Mobile GIS"

After successfully packaging the "RU Mobile GIS", the windows installer can be found at ...\RU_Mobile_GIS\Release directory. To deploy, copy the content of this directory onto the destination system.

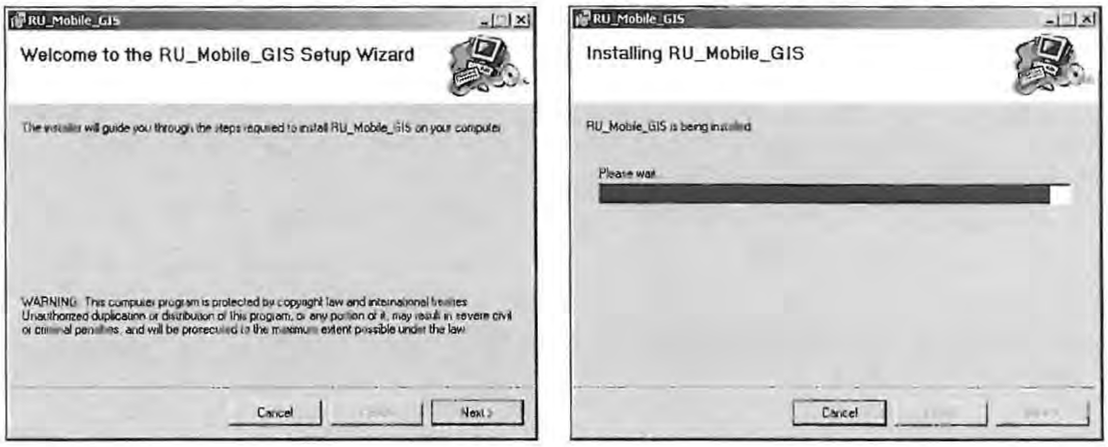

Release
File Edit View Favorites Tools Help
Back Search Folders
Links Google yahoo Master page delay pool Test test2

Address: ootProject\demos\Mobile\SetupProjects\RU_Mobile_GIS\Release

InstMsiA.Exe
InstMsiW.Exe
RU_Mobile_GIS.msi
Setup.Exe
Setup.Ini

FIGURE 4.4.2-13 – CONFIGURATION MANAGER

Double click the "setup.exe" icon to start up the mobile Web application installation wizard. Follow the instructions to complete the installation. The application, when unpacked becomes a virtual directory on the destination system.



Welcome to the RU_Mobile_GIS Setup Wizard

The installer will guide you through the steps required to install RU_Mobile_GIS on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Installing RU_Mobile_GIS

RU_Mobile_GIS is being installed.

Please wait...

FIGURE 4.4.2-14 – INSTALLING "RU MOBILE GIS"

When the installation process has finished the *"RU Mobile GIS"* can be accessed through the Internet using "http://hostname(fully qualified URL of the destination system)/RU_Mobile_GIS", in the same way as accessing any other Web applications.

4.5 Design the backend database component

4.5.1 Mobile .NET database Connectivity – ADO.NET

4.5.1.1 ADO.NET introduction

Being a .NET Mobile Web Application, the *"RU Mobile GIS"* uses the ADO.NET model to maintain the connection and operation of the backend database. The "ADO" in ADO.NET stands for ActiveX Data Object, a part of the .NET framework architecture.

As we know, the traditional data connection and access relies primarily on a connection-oriented, two-tier model. In the fast growing distributed Web-based application environment this traditional approach becomes less and less flexible and cannot provide good support to the multi-tier data access architecture. ADO.NET provides a new concept in terms of accessing data source. It is a disconnected accessing mechanism, using XML as a medium for moving data between tiers.

4.5.1.2 ADO.NET data components

There are two central components of ADO.NET that facilitate data access through data manipulation [Please see figure 4.5.1-1].

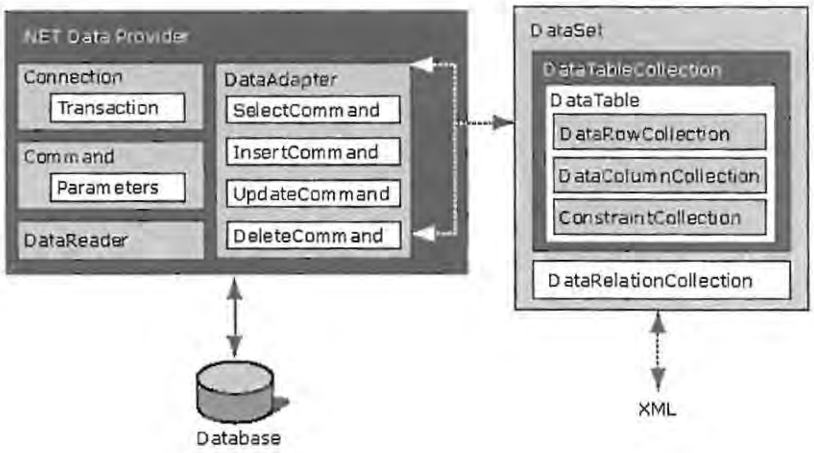
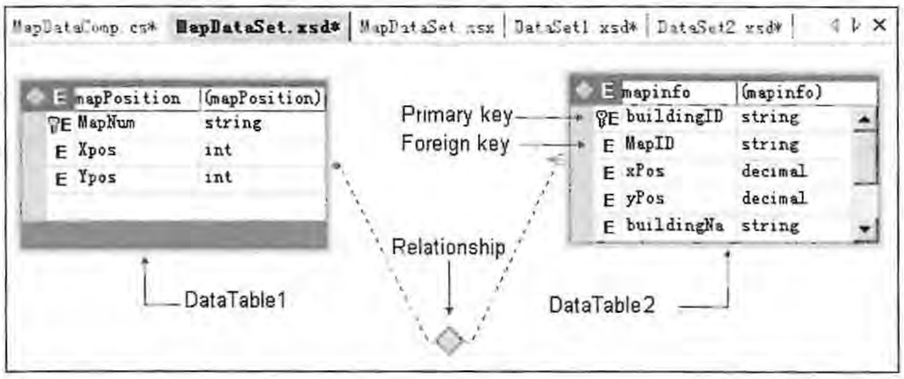


FIGURE 4.5.1-1 – ADO.NET ARCHITECTURE [MICROSOFT]

DataSet

DataSet is the key component of the disconnected accessing mechanism of ADO.NET architecture. Using XML as the data storage medium, the DataSet is independent of any data source. It can therefore be used to accommodate heterogeneous data sources.

The DataSet may have one or more DataTable objects. A DataTable contains rows and columns, and gives the programmer the ability to specify a primary key, foreign key, and relationships between tables. The diagram below shows the relationship of "MapDataSet" and its DataTables in Visual Studio.NET devolvement environment.


MapDataComp.cs* | MapDataSet.xsd* | MapDataSet.xsx | DataSet1.xsd* | DataSet2.xsd*
Primary key
Foreign key
Relationship
DataTable1
DataTable2

E mapPosition	(mapPosition)
E MapNum	string
E Xpos	int
E Ypos	int

E mapinfo	(mapinfo)
E buildingID	string
E MapID	string
E xPos	decimal
E yPos	decimal
E buildingNa	string

FIGURE 4.5.1-2 – USING DATASET IN VISUAL STUDIO.NET

Data provider

Data providers are used for data manipulation of read-only and forward-only data sources. ADO.NET Framework provides two options for connecting two databases:

1) OLE DB.NET data provider.

OLE DB.NET Data provider is in the System.Data.OleDb namespace[17], and is a set of components including the "OleDbConnection", "OleDbCommand", "OleDbDataReader" and "OleDbDataAdapter" objects.

2) SQL Server.NET data provider

SQL Server.NET Data Provider is in the System.Data.SqlClient namespace, and is a set of components including the "SqlConnection", "SqlCommand", "SqlDataReader" and "SqlDataAdapter" objects.

4.5.1.3 Using ADO.NET

In the *"RU Mobile GIS"* both the "MapDataComp" class and the "AccountDataComp" class use the SQL Server.NET Data Provider to connect directly to the Microsoft SQL database.

[17] "namespace" is a concept used to group classes that provide certain functions together separating them from other classes and namespaces. It is similar to the term "package" in Java.

```
using System.Data;
using System.Data.SqlClient;
...
namespace RUGIS
{
    public class MapDataComp : System.ComponentModel.Component
    {
        private SqlConnection sqlConnection1;
        private SqlDataAdapter sqlDataAdapter1;
        private SqlDataReader sqlReader1;
        private SqlCommand sqlCommand1;
        ...
        private void InitializeComponent() {
            ...
            sqlDataAdapter1 = new SqlDataAdapter();
            sqlCommand1 = new SqlCommand();
            sqlConnection1.ConnectionString = "data source =CSDANNY\\NETSDK;
            initial catalog=RHODESGIS;integrated security=SSPI; per" +"sist security
            info = True; workstation id=CSDANNY; packet size=4096";
            ...
        }//InitializeComponent
        ...
        public void buildingName(string mapID) {

            sqlCommand1.Parameters["@MapID"].Value = mapID;
            ...
            sqlCommand1.CommandText = "SELECT buildingName FROM
            mapinfo WHERE(MapID = @MapID)";
            sqlConnection1.Open();
            sqlReader1 = sqlCommand1.ExecuteReader();
            while(reader.Read())
            {
                resultItems.Add(sqlReader1.GetString(0));
            }
            sqlConnection1.Close();
            sqlReader1.Close();
            }
        }//buildingName
        ...
}
```

CODE 4.5.1-1 – MAPDATACOMP

In the previous code snippet:

1) The "sqlConnection1" object is used to setup the connection to the SQL Server.

2) The "sqlCommand1" object stores procedures to be executed later that retrieve, update, and delete data from the SQL database.

3) The "sqlReader1" provides the actual data stream from the database.

4) The "sqlDataAdapter1" object bridges the DataSet object and the data source. It uses "sqlCommand1" objects to execute SQL commands at the data source. It is responsible for both loading the DataSet with data, and modifying changes made to the data in the DataSet back in the SQL database.

4.5.1.4 Using Visual Studio.NET to set up database connection

Visual Studio.NET provides a set of advanced utilities to set up database connections and generate data components. The Server Explorer displays all available servers, their databases, data tables, and even data columns in a tree structure. To connect to a database, simply select a specific data element from the Server Explorer and drag and drop it into the Design Window. Visual Studio.NET generates the relevant data components automatically [See figure 4.5.1-3].

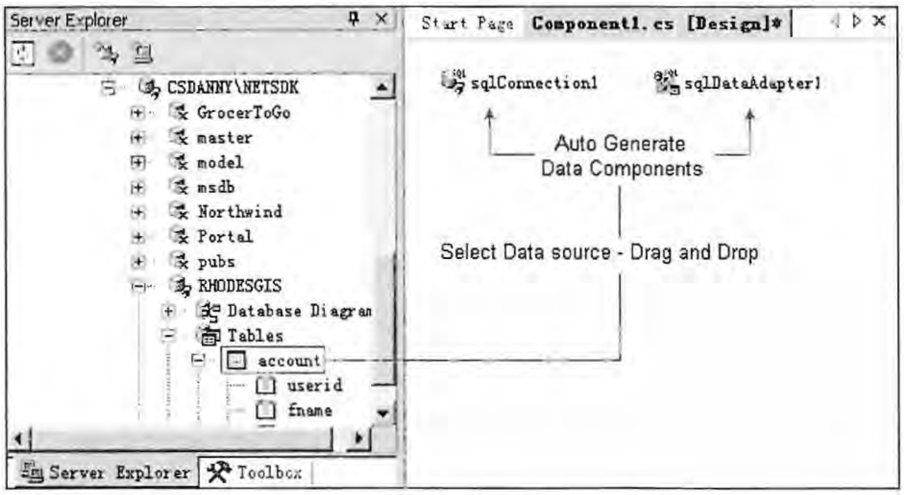
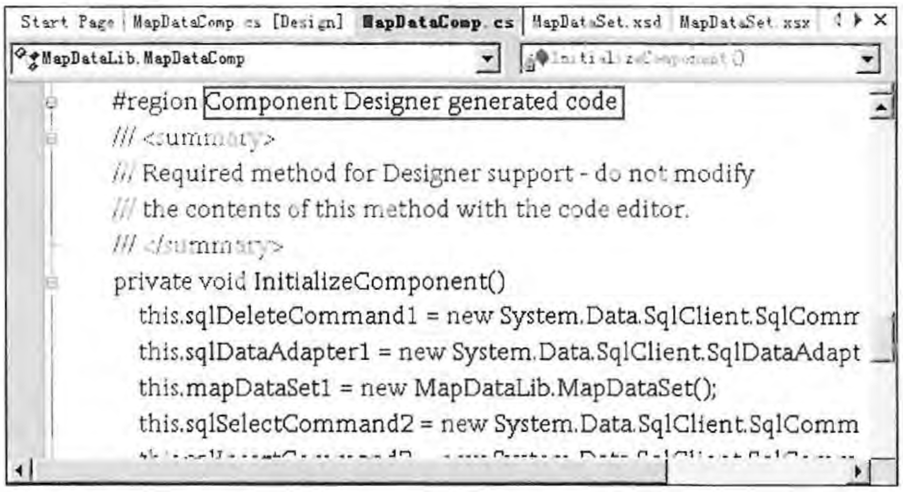


FIGURE 4.5.1-3 – SETUP DATABASE CONNECTIVITY IN VISUAL STUDIO.NET

In addition, Visual Studio.NET generates the code automatically, including declarations of data objects and initialisation of instances [Please see figure 4.5.1-4].

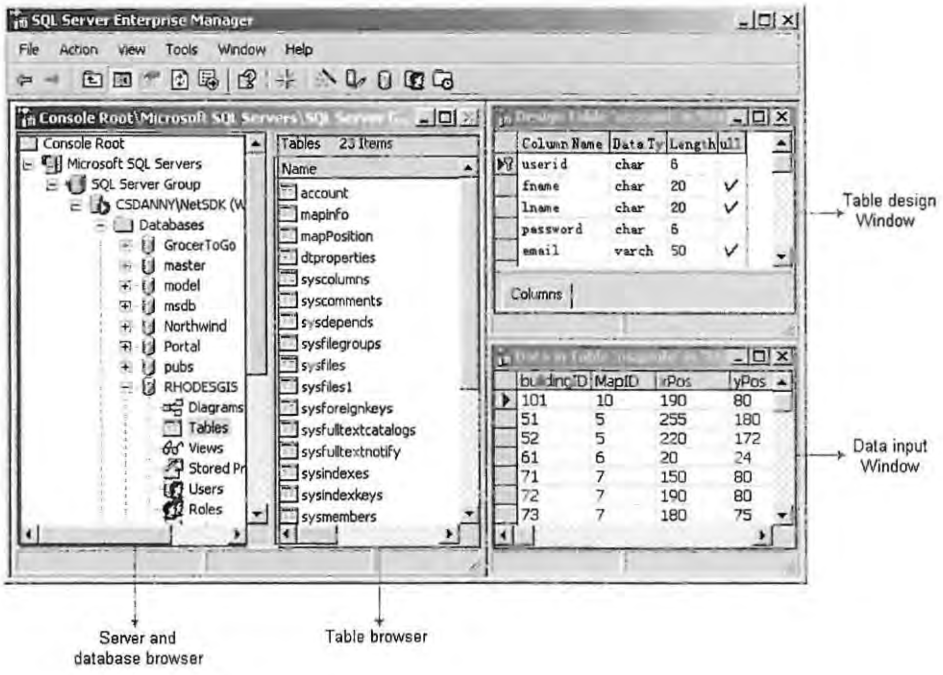


```
#region Component Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
    this.sqlDeleteCommand1 = new System.Data.SqlClient.SqlComm
    this.sqlDataAdapter1 = new System.Data.SqlClient.SqlDataAdapt
    this.mapDataSet1 = new MapDataLib.MapDataSet();
    this.sqlSelectCommand2 = new System.Data.SqlClient.SqlComm
```

FIGURE 4.5.1-4 – CODE AUTO GENERATION

4.5.2 Microsoft SQL Server 2000

The Microsoft SQL Server 2000 is used to manage data for the *"RU Mobile GIS"*. It offers a set of fully graphical utilities, such as SQL Server Enterprise Manager [Please see the figure below]. This means that even developers with "zero knowledge" about SQL programming are able to create database, manipulate tables, input data and so forth.



Column Name	Data Ty	Length	ull
userid	char	6	
fname	char	20	✓
lname	char	20	✓
password	char	6	
email	varch	50	✓

buildingID	MapID	xPos	yPos
101	10	190	80
51	5	255	180
52	5	220	172
61	6	20	24
71	7	150	80
72	7	190	80
73	7	180	75

FIGURE 4.5.2-1 – SQL SERVER ENTERPRISE MANAGER

4.5.3 RhodesGIS database

"RhodesGIS" is the database created for this application. There are three tables involved: the "accounts" table, the "mapinfo" table and the "mapPosition" table.

4.5.3.1 The "accounts" table:

The "account" table holds user information such as usernames, passwords and email addresses. It is queried when the "login()" method is invoked, and is updated when the "register()" method is called.

Accounts					
Field	**Type**	**Null**	**Key**	**Default**	**Extra**
UserID	char(6)	NO	YES	NULL	
Fname	char(20)	YES		NULL	
Lname	char(20)	YES		NULL	
password	char(6)	NO		NULL	
Email	varchar(50)	YES		NULL	

TABLE 4.5.3-1 – ACCOUNTS TABLE

4.5.3.2 The "mapinfo" table:

The "mapinfo" table contains a detailed description of each building within a certain map. The description includes the building location, building information and a Web page URL. It is queried when the "listDataBinding()" method, "getBuildingName()" method and "getBuildingDetail()" method are invoked.

mapinfo					
Field	**Type**	**Null**	**Key**	**Default**	**Extra**
buildingID	char(3)	NO	YES	NULL	
mapID	char(2)	NO	YES	NULL	
Xpos	int(4)	NO		NULL	
Ypos	int(4)	NO		NULL	
buildingName	char(20)	NO		NULL	
buildingInfo	varchar(500)	YES		NULL	
WebPage	varchar(50)	YES		NULL	

TABLE 4.5.3.2 – MAPINFO TABLE

4.5.3.3 The "mapPosition" table:

The "mapPosition" table contains values of the map coordinates. It is queried when the "displayLocation()" and "jumpLocation()" methods are invoked.

MapPosition					
Field	**Type**	**Null**	**Key**	**Default**	**Extra**
mapID	char(2)	NO	YES	NULL	
Xpos	int(4)	NO		NULL	
Ypos	int(4)	NO		NULL	

TABLE 4.5.3-3 – MAPPOSTION TABLE

4.6 Chapter review

In this chapter we investigated the development life cycle of the .NET mobile Web applications in a real working environment. First we took a look at the software development environment for building the .NET mobile Web application, and then we introduced the Mobile ASP.NET. Finally, we examined the development and deployment of the "Rhodes University Mobile Geographic Information System", the middle tier Web component development and the backend database design.

Chapter 5: J2ME vs. .NET – Development Approaches

– *"Work joyfully and peacefully, knowing that right thoughts and right efforts will inevitably bring about right results"* [Ferguson, Derek].

This chapter reviews the development environments of both the J2ME and the .NET mobile Web applications that we introduced in the previous two chapters. Detailed comparisons are made between the two approaches, and then generalized guidelines are drawn on how to rapidly and efficiently develop highly portable and easily extensible mobile Web applications with high performance, good usability and strong reliability.

5.1 J2ME vs. .NET – An overview

Plenty of aspects and facts have been discussed in chapter three and four with regards to developing mobile Web applications using both J2ME and .NET mobile environment. In this section, one-on-one comparisons are drawn between a variety of features of development mobile Web application under the J2ME and the .NET environments.

Table 5.1.1-1 summarizes the differences between using the J2ME and .NET mobile platforms to develop mobile Web applications.

Mobile Web Application & Its Development Environment		
	J2ME	**.NET Mobile**
General		
Dependency	Platform independent and language specific	Platform dependent but language unspecific
Language support	Java	C#, VB.NET, JScript...
Application Type	Java Application (MIDlet)	Web page (ASP.NET page)
Deployment	Client-side and server-side	Only server-side deployment
Portable	Yes	No
Mobile Device		
Runtime requirement	Java Virtual Machine	WML, HTML or cHTML support
Local storage	Required	Not necessary
Client-side data processing ability	Supported	Not supported
Event Handling	Client-side	Server-side

Mobile Web Application & Its Development Environment (Continued)		
	J2ME	**.NET Mobile**
Web Server		
Web Server	Apache, Sun, Netscape, IBM, W3C...	Microsoft IIS
Technologies	Java Servlet, JSP, CGI, PHP...	ASP.NET
Role of Web Server	Service provider	Generate pages, translate Markup language
Development Tools		
Operating System	Unix, Linux, Solaris, Windows...	Microsoft Windows
IDE	SunONE Studio Mobile Edition	Microsoft Visual Studio.NET
Application SDK	J2ME Wireless Toolkit	Mobile Internet Toolkit
SDK Integration	Standalone	Integrated with VS.NET
GUI Design	Text editor	Graphical Designer
Events wired up	Explicitly	Automatically
Exception handling	Force declare of handle	Optional
HTTP Connection	Programmatically set up connection	Browser default connection
Data Connection	Programmatically set up connection	Using server browser
Help System	Static APIs	Dynamic Help System
Support		
Standard	Open	Proprietary
Developers and Supporters	SUN, Motorola. Ericsson, Nokia, Siemens, Sharp, Philips, DoCoMo ...	Microsoft

TABLE 5.1.1-1 – J2ME VS. .NET MOBILE

5.1.1 General comparisons

5.1.1.1 Platform dependency

J2ME:

As mentioned in section 3.2.1.2, because of the machine independency of Java, the development and deployment of J2ME mobile Web applications are platform independent.

.NET:

On the contrary, .NET mobile Web applications can only be developed and deployed exclusively in a Microsoft environment.

Comparison and Evaluation:

Platform independency and high portability make Java preferable over other languages and platforms. And there is no exception for the Java 2 Platform Mobile Edition (J2ME). Choosing J2ME as the development environment enables developers to create mobile Web applications in any given Operating System, and deploy applications to any targeting device and Operating System without code modification. The J2ME development environment is superior to the .NET mobile Web application development environment in terms of its portability (there are some non-commercial efforts underway to enable .NET clients to run on platforms other than Microsoft operating systems, such as Mono[18], but these are not yet commercially viable).

5.1.1.2 Language support

J2ME:

Java is the only language that can be used to build J2ME mobile Web application.

.NET:

.NET gives developers a rich set of languages to choose from for the implementation of mobile Web applications, which include C#, C++, VB.NET, JScript and so forth.

Comparison and Evaluation:

In the .NET development environment all languages are complied into Microsoft Intermediate Language (MIL), which then relies on the Common Language Runtime (CLR) - a huge collocation of APIs. The .NET multi-language choices are, therefore, only choices of language syntax, not of run-time libraries. However, this is still a very powerful feature. Under the .NET mobile Web application development environment, developers can choose their favorite programming language to create mobile Web applications. Consequently, more programmers are attracted and involved in the development of applications since they do not have to learn another language in order to use the environment. In the case of J2ME, only Java programmers are able to develop J2ME mobile web applications.

[18] The Mono Project is an open development initiative sponsored by Ximian that is working to develop an open source, Unix version of the Microsoft .NET development platform. Its objective is to enable Unix developers to build and deploy cross-platform .NET Applications. The project will implement various technologies developed by Microsoft that have now been submitted to the ECMA for standardization.

5.1.1.3 Application type and executing speed

J2ME:

As introduced in section 3.3, J2ME mobile Web applications are called MIDlets. They are Java applications that run on Java enabled mobile devices. As shown in section 3.3.5.2, J2ME is a reduced set of Java APIs that is designed specifically for mobile devices. It normally only takes up a few kilobytes of physical storage. The J2ME runtime environment is different from the Java standard version. The lightweight implementation, the use of a "Preverifier" and the reduction of the "byte code verifier" make the J2ME mobile Web application run much faster [Please refer to table 5.1.1-2], and consume less system resources In addition frequent HTTP requests are not required - only necessary data and information is exchanged between mobile device and Web server This reduces the latency the HTTP round-trip results in, and increases application execution speed dramatically.

Speed Up Over Traditional Java Runtime Interpreter		
Benchmark Tools	**Traditional Interpreter (HS)**	**Mobile Interpreter**
Richards	0%	12%
Deltablue	0%	30%
Pentominoes	0%	-7%
Mandelbrot	0%	13%
BenchPress	0%	5%
SciMark Composite	0%	7%
CaffeineMark	0%	15%
jBYTEmark, Integer Index	0%	-6%
JBYTEmark, FPIndex	0%	24%
Average		**10%**

Source: [Bak, Lars]

TABLE 5.1.1-2 – SPEED UP OF MOBILE JAVA INTERPRETER

.NET:

.NET mobile Web applications are actually dynamic Web pages (ASP.NET Web Pages) that run on Web servers and are accessed by mobile devices. There is no intensive client-side computation required for running .NET mobile Web applications, but frequent HTTP requests are necessary. The consequences of making frequent HTTP requests are discussed below.

Comparison and Evaluation:

The execution speed of J2ME mobile Web applications relies mainly on the processing speed of the mobile devices on which applications are running. The execution speed of .NET mobile Web applications depends largely on the network speed.

Usually the bandwidth of a wireless network is much smaller than a fixed-wire connection. Table 5.1.1-3 lists the bandwidth and latency of three wireless networks.

Wireless Network Bandwidth and Latency			
	Bandwidth (kilobits/second)	Typical Initial Connection Set- up Delay	Estimated Latency (HTTP round- trip)
GSM Data	9.6 - 14.4 kbps	5 - 10 seconds	2 - 3 seconds
HSCSD[19]	up to 43 kbps	5 - 10 seconds	2 - 3 seconds
GPRS	5 - 50 kbps	none 3 - 5 seconds at power-on or 1st session	3 - 6 seconds when network is congested, can be more

Source: [Price, David]

TABLE 5.1.1-3 – WIRELESS NETWORK BANDWIDTH AND LATENCY

Consider the registration and login Module for example. In the J2ME module, there are three steps to register a new user (Time spent for user input is not taken into account).

1) Fill in the personal details and click next;

Items of personal information are stored in local variables after which the next page is displayed. No communication is established between client and server. Approximate execution time: 0.1 second.

2) Fill in the contact detail and click next;

Items of contact information are stored in local variables after which the next page is displayed. No communication is established between client and server. Approximate execution time: 0.1 second.

[19] HSCSD – High Speed Circuit Switched Data, GPRS – General Packet Radio Service

3) Choose a username and password and create the account;

An HTTP connection channel is created. Username and password along with personal and contact details are sent to the server. The information is displayed based on the server feedback. Approximate execution time: 2-3 seconds.

Similarly, the user has to follow the same three steps to complete registration using a .NET mobile Web application.

1) Fill in the personal details and click next.

The user's action is communicated to the server, while items of personal information are stored using a selected server-side session state management technology. The server sends back the next page, which the client device then displays. Approximate execution time: at least 2-3 seconds.

2) Fill in the contact detail and click next.

The user's action is communicated to the server, while items of contact information are stored using a selected server-side session state management technology. The server sends back the next page, which the client device then displays. Approximate execution time: at least 2-3 seconds.

3) Choose a username and password and create the account.

The user's action is communicated to the server, as well as the username and password. The server then creates the new user and sends back the confirmation page. The client device then displays the new page. Approximate execution time: at least 2-3 seconds.

A new user account can be created under J2ME mobile Web application in around 3 seconds with the HTTP connection only needing to be established once, while it takes a .NET mobile Web application 7 to 9 seconds and three HTTP round-trips to complete the same transaction.

Given these facts and the table 5.1.1-3, we can see the HTTP round-trip latency is the obvious bottleneck for .NET mobile Web applications Therefore, unless a high quality network environment is available, the execution speed of.NET mobile Web applications is slower than J2ME mobile Web applications.

5.1.2 Mobile device comparisons

5.1.2.1 Runtime requirement

J2ME:

As a Java application, J2ME relies on the Java Virtual Machine (JVM) to run. Therefore a JVM has to be implemented and deployed on each of the mobile devices.

.NET:

As was discussed in section 4.4.1, .NET mobile Web applications (mobile ASP.NET pages) can be interpreted into different Markup languages, including WML, HTML, and cHTML. Hence specific types of Web browsers, which can use .NET applications, should be placed on mobile devices.

Comparison and Evaluation:

To equip mobile devices with Java support, chips designed specifically for Java byte code execution should be embedded inside each of mobile device by the mobile device manufacturer. This is problematic since not all mobile devices on the market support Java and J2ME technologies. Since .NET mobile Web applications require no hardware implementation or special deployment of runtime environments, they are adaptable for a wider range of mobile devices.

The fast growing market share and the popularity of Java enabled mobile devices is gradually reducing the gap. In 2001 alone the number of Java mobile device users grew from 1.6 million to 10.48 million in Japan [NTT DoCoMo], a trend that continues. There is thus reason to believe that in the near future the majority of mobile devices will support Java and J2ME technologies. With this mobile device support there will no longer be this gap between J2ME and .NET mobile Web applications.

5.1.2.2 Local storage

J2ME:

In section 3.3.5.4 we introduced the idea of deploying a J2ME mobile Web application. J2ME mobile Web applications are downloaded across the Internet and stored in mobile devices.

.NET:

The client-side component of .NET mobile Web applications is a browser. The entire application is executed on the server side; hence no local storage is required.

Comparison and Evaluation:

The number of J2ME Web applications that can be installed on a mobile device is limited by the local storage capacity. Once the local storage is used up no more applications can be installed. .NET mobile Web applications reside in Web servers. There is therefore no limit to the number of applications that a mobile device can run, even if its local memory is limited. Some mobile devices, such as Pocket PCs, use shared system memory for both application storage and execution. Consequently, the more memory spent on application storage the less is available for application execution. In another words, such applications are likely to run much slower, since system memory is a critical factor in application execution speed. Mobile devices are normally equipped with limited amounts of memory, and some low-end mobile devices have only several kilobytes of memory. Therefore, with any given amount of system memory, more .NET mobile Web applications can be accessed than J2ME mobile Web applications.

5.1.2.3 Client-side data processing ability and event handling

J2ME:

For J2ME mobile Web applications, data and user actions can be processed and handled on the client side.

.NET:

Every single user event is passed to the Web server and handled remotely; no data can be processed on the client side.

Comparison and Evaluation:

Today's mobile devices that run mobile Web applications are no longer simple terminals that only display text messages, receive and initialise voice communication. They are devices equipped with processors powerful enough to process a wide range of data. For example, a mobile device like Pocket PC has a 206MHz StrongARM 1110 processor, which is fast enough to process multimedia data. J2ME Web applications make full use of the client-side computational power to handle events and process data on the client side. This shares the server-side computational overhead, thus improving the efficiency of the entire system. .NET mobile Web applications rely on server-side event handling and data processing and do not exert the processing power of mobile devices. Thus for .NET mobile Web applications network throughput is a critical factor while for J2ME mobile Web applications the processor speed is essential. [For a detailed analysis on the two different models, refer to section 5.3.1].

5.1.3 Web server comparisons

5.1.3.1 Web server

J2ME:

The server-side components of J2ME mobile Web applications can be deployed onto a wide range of HTTP Web servers. Please refer to Table 5.1.1.3-1.

Java Server-Side Technologies Enabled HTTP Servers		
Vendor	**Product**	**URL**
Apache	Apache Web Server	www.apache.com/pws1410.php3
Lotus	Domino Go Web Server	www.lotus.com/home.nsf/welcome/domino
IBM	Internet Connection Server	www.as400.ibm.com/developer/ebiz/
	Visual Age WebRunner	www.ibm.com/software/ad/webrunner/
	WebSphere Application Server	www-4.ibm.com/software/webservers/
Netscape	Netscape Enterprise Server	home.netscape.com/enterprise/v3.6/
O'Reilly	Website Professional	www.oreilly.com/catalog/webpro2/
Sun	Sun Web Server	www.sun.com/products-n-solutions/
	Java Web Server	www.sun.com/software/jwebserver/
W3C	Jigsaw HTTP Server	www.w3.org/Jigsaw/
Web Easy	WEASAL	www.webeasy.com/products/weasel.htm
Zenus	Zenus Web Server	www.zenus.com/pr.html
Live	JRun	www.jrun.com/Products/Jrun/

Source: [LEE, Raymond Shu-Tak]

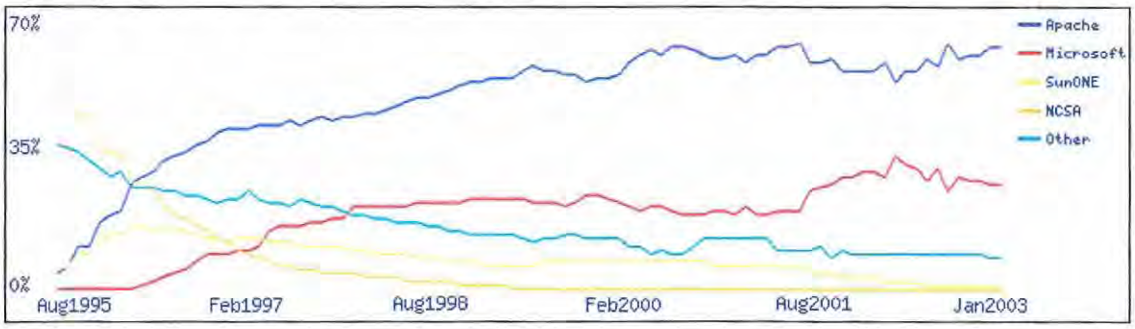
TABLE 5.1.3-1 – JAVA ENABLED HTTP SERVERS

.NET:

.NET mobile Web applications can only be deployed onto Microsoft Internet Information Servers (IIS) with .NET framework installed. There is no other Web server available thus far.

Comparison and Evaluation:

Figure 5.1.3-1 and table 5.1.3-2 show the numbers and percentages of HTTP Web servers that are current running worldwide.



Source: [Netcraft]

FIGURE 5.1.3-1 – HTTP SERVER PERCENTAGE (JAN 2003)

Active Sites		
Developer	**January 2003**	**Percentage**
Apache	11178715	66.42%
Microsoft	4172101	24.79%
Zeus	261652	1.55%
SunONE	233105	1.39%

Source: [Netcraft]

TABLE 5.1.3-2 – ACTIVE HTTP SERVER NUMBERS AND PERCENTAGES (JAN 2003)

As we can see, nearly 76%[20] of HTTP Web servers support Java server-side Technologies. Fewer than 20%[21] of servers can host .NET mobile Web applications. Obviously J2ME mobile Web applications have more server-side support then .NET mobile Web applications. In Chapter Three and Chapter Four, the architectures of J2ME and .NET mobile Web applications were introduced, both of which use a multi-tier model. To deploy a mobile Web application successfully the middle tier component needs to be configured on an HTTP server. Because of the runtime environment requirement, the server-side component of a .NET mobile Web application can only be deployed onto HTTP servers that have .NET framework support. In another words, .NET mobile Web applications cannot be deployed on most legacy HTTP servers. In contrast with this most existing HTTP servers offer good support for Java.

5.1.3.2 Server-side technologies support

J2ME:

J2ME mobile Web applications can be integrated with various server-side technologies such as Java Servlet, Java Server Pages, Common Gateway Interface, PHP and so forth.

[20] All Http servers except Microsoft IIS in table 5.1.3-2 supprot Java technologies so the pecetage is around 76.

[21] Within this 24.79%, only IISs with .NET framework installed are able to host .NET mobile Web applications. Therefore the actual percentage of Web servers that support .NET mobile Web applications is much less than 20%.

.NET:

Mobile ASP.NET is the only server-side technology that supports .NET mobile Web applications.

Comparison and Evaluation:

Compared with a .NET mobile solution, J2ME provides a more flexible solution that adapts itself to more server-side technologies. For example, Java Servlet is used as the server-side component of the J2ME mobile Web application – *"Mobile Meal Booking System"*. The Java Servlet acts as a data processor that handles HTTP requests from the client application, in this case a MIDlet, and sends HTTP responses back. In this scenario the Servlet can be substituted by other server-side technologies as long as they are capable of manipulating HTTP requests and responses. Consequently, J2ME client mobile Web applications are able to interact with existing server-side services and applications developed using legacy server-side technologies, without much effort or code modification.

5.1.4 Development tools comparisons

5.1.4.1 IDE

J2ME:

The official J2ME mobile application Integrated Development Environment is the SunONE studio.

.NET:

The IDE for the development of .NET mobile Web applications is Visual Studio.NET.

Comparison and Evaluation:

Visual Studio.NET is an efficient, well-designed, Integrated Development Environment with all the necessary features for development of mobile Web applications with usability imbedded at the heart of the design. It provides developers with rich sets of utilities for the rapid development of mobile Web applications. From section 4.4.2 and table 5.1.1-1 we can see that Visual Studio.NET has approximately twice as many features and utilities than the J2ME application development environment. Using Visual Studio.NET, the *"RU Mobile GIS"* was created within a week, while the *"Meal Booking System"* was completed in approximately two weeks under the J2ME development environment. Compared with Visual Studio.NET, other IDEs such as SunONE studio still need significant improvements to reach the same level as Visual Studio.NET. With the help of Visual Studio.NET, mobile Web applications can be developed much more efficiently.

5.1.4.2 Application SDK and its integration

J2ME:

J2ME Wireless Toolkit (JWTK) is a stand-alone SDK for the development of J2ME mobile Web applications. It comes with a set of mobile device emulators and application performance test utilities.

.NET:

The Microsoft Mobile Internet Toolkit (MMIT) is the SDK used to develop .NET mobile Web applications. It is integrated with Visual Studio.NET seamlessly. When developing .NET mobile Web applications developers work with the Visual Studio.NET IDE. The invocation and cooperation between the IDE and the SDK are transparent to the developer.

Comparison and Evaluation:

As was examined in section 3.3 and section 4.4, JWTK provides excellent application monitoring and testing facilities while MMIT gives a better development environment. For example, JWTK offers a real time monitor that analyses the memory usage of each object while running a mobile Web application..NET only gives an HTML log for the programmer to examine execution states when the application is terminated. The lack of real-time monitoring makes MMIT incapable of observing and analysing the entire execution life cycle of a given mobile Web application.

5.1.4.3 GUI design

J2ME:

J2ME develop environment has no GUI designer. GUI components of J2ME mobile Web applications are normally composed using a text editor.

.NET:

Introduced in section 4.4.2, Virtual Studio.NET offers developers an outstanding GUI design environment for the development of mobile Web applications. GUI components can be designed in a drag-and-drop manner.

Comparison and Evaluation:

The graphical design environment can reduce the GUI design complexity and increase the application development efficiency dramatically. Lack of a graphical GUI designer makes the creation of GUI components in a J2ME mobile Web application a complex job, one prone to errors.

5.1.4.4 Coding

J2ME:

Every line of code is hard-coded, from the HTTP connection initialisation to the database connection setup; from event declaration to event delegation wire-up.

.NET:

When writing a .NET mobile Web application, a large amount of code can automatically be generated, such as database connection setup, event wire-up, component declaration and initialisation.

Comparison and Evaluation:

Using the .NET mobile application development environment means that developers are able to concentrate on writing application business logic instead of interfaces. Therefore, applications can be developed much faster and more efficiently. The rapid prototyping of applications is greatly supported.

5.1.4.5 Help system

J2ME:

The J2ME provides a static help system - a separate package that contains APIs and sample application code.

.NET:

In section 4.4.2.2 we introduce the Visual Studio.NET help and API system, which is integrated with the IDE; it dynamically retrieves and displays information on a desired topic by simply highlighting the keyword.

Comparison and Evaluation:

Compared to the J2ME help system, the .NET help system's topics and sample code are organized and presented in a more meaningful way and can be found much faster. When developing an application, developers normally spend considerable amounts of time working with language APIs, the help system and sample code. A well designed and organized help system can reduce time spent on finding information and therefore speed up the development of the application. The .NET dynamic help system is easier and more efficient to use than the J2ME static help and API system.

5.1.5 Industry support comparisons

5.1.5.2 Developers and supporters

J2ME:

J2ME is supported not only by Sun Microsystems but also by a large number of major IT, telecommunication, electronic companies and organizations such as Motorola, Ericsson, Nokia, Siemens, Sharp, Philips and DoCoMo.

.NET:

The .NET mobile solution was developed by Microsoft. As of yet no other company has announced official support of the .NET mobile solution.

Comparison and Evaluation:

Good industry support is critical to the success of a technology. As discussed in section 2.2.4, .NET is a set of Microsoft software technologies. It is a long term strategic initiative launched by Microsoft. Mobile .NET forms an essential link in the whole Microsoft .NET chain. Microsoft, as the biggest software company in the world, have confidence in their research, marketing and competitive strength. Money and effort will be put into the development, spreading and advertisement of Mobile .NET continuously. J2ME, on the other hand, is an open standard, and with the support of a wide range of industry giants it proves to remain a significant competitor for Microsoft in the foreseeable future in the face of the might of Microsoft's hold over the microcomputer market.

5.2 Mobile devices investigation

From the experience gained through the development of both the J2ME and the .NET mobile Web applications, it is apparent that knowing the platform we are dealing with and understanding the limitations of the devices we are programming for is very helpful in developing successful applications.

5.2.1 General limitation of mobile devices

Although mobile device capabilities differ from one to another, they all share common limitations.

5.2.1.1 Mobile devices limitations

1) Limited processing power:

Mobile devices are usually run on battery power, and so cannot support high-end processors, which normally have high power consumption.

2) Small screen size

Mobile devices, especially cellular phones, have very limited display capabilities. More specifically, some of them do not have colour, do not support images and can only display two or three lines of text.

3) Insignificant local storage capacity:

Most mobile devices have only several megabytes or even a few kilobytes of memory for information storage

4) They have restricted input mechanisms

Most mobile devices, typically cellular phones, are only equipped with an alphanumeric keypad.

5) Low bandwidth and unstable network connection

The underlying networks that mobile devices rely on are normally only capable of running at a "kilobits" speed level. Being wireless, they are subject to constant disconnections.

6) Uncontrollable application environment

There are hundreds of manufacturers that follow a vast range of standards to produce mobile devices. Thus, the mobile devices available today run different protocols. Taking the Web protocol for example, there are so many variations: WML, HTML, cHTML, XHTML and so forth.

5.2.1.2 Development guideline – How to avoid these limitations?

Unfortunately these are natural limitations of mobile devices and are unavoidable. There are, however, guidelines to follow that help when designing, developing and deploying J2ME and .NET mobile Web applications.

1) .NET mobile Web applications assign the data processing and computation completely to the server. When designing a J2ME mobile Web application, on the other hand, one should place serious computation intensive tasks on the client-side to avoid slowing down the entire application.

2) When designing a mobile Web application, whether J2ME or .NET, use short, comprehensive labels instead of meaningful, verbose sentences. For example, when prompting for user input, use "name" instead of "Would you please fill your name in the text box below?"

3) A J2ME mobile Web application that is deployed onto a mobile device includes a java Archive (JAR) file that contains complied source code, application resources such as pictures, and a Java Application Descriptor (JAD). The JAR file is stored in the mobile device and loaded for execution. Make the size of a J2ME mobile application as small as possible to increase the speed of downloading and executing and to make space available for other mobile applications. Since a .NET mobile application is made up of mobile Web pages stored on the server side and displayed on the client, its size is not as crucial as that of a J2ME mobile Web application. For both J2ME and .NET mobile Web applications, do not include large binary files such as graphics or sound unless absolutely necessary. When using binary files rather choose a compressed format than uncompressed. For example, use ".png" instead of ".bmp" and use ".mid" instead of ".wav". The result of this is that mobile Web pages will be loaded much faster for .NET mobile Web applications and JAR files will be smaller for J2ME mobile Web applications.

4) Reduce the chances of unnecessary user input - collect only essential information from the user. This rule applies to the design and development of both J2ME and .NET mobile Web applications.

5) As has been addressed several times, HTTP communication under a real wireless Internet environment is time-consuming. To increase the mobile Web applications' execution speed for J2ME, only set up communication between client and Web server when necessary. In the case of .NET, when communicating with the server, send all data as a whole once instead of separating the information into smaller units and sending them individually.

6) Protocol coverage and support should be maximized when choosing J2ME and .NET mobile Web application development approaches and tools.

5.2.2 Mobile device diversity

As already addressed, mobile devices differ from one another in hundreds of ways. Some have colour support while others have not, some can display graphics while others cannot, and some devices can place phone calls while others cannot. Consequently, when developing mobile applications the diversity of mobile devices must be considered.

5.2.2.1 J2ME approach

J2ME uses a well-designed approach to manage device diversity. It categorises devices into groups using configurations and profiles. For example the Connected Limited Device Configuration (CLDC) / Mobile Information Device Profile (MIDP) is used to create the *"Mobile Meal Booking System"* on cell phones and PalmOSs while the Personal Digital Assistant Profile (PDAP) is used to build application targeting PDAs. Every configuration and profile differs from each other one in terms of functionalities.

Advantages

1) By classifying mobile devices into different groups, applications can be developed in a standardised manner, thereby boosting the application's portability.

2) Application functionality and design purpose are clear. Special characteristics of certain devices or categories can be implemented and enhanced.

Disadvantages

1) The adoption and standardisation of new technologies can be drawn out over a long period of time.

5.2.2.2 .NET approach

The .NET mobile solution adopts a completely different approach to handling differences between devices. .NET uses filters to isolate device-specific function modules, and classifies devices individually. A filter is a section of code that can be used to test device capability. Filters are registered in the "Web.config" file and support customisation.

Advantages

1) Allows high customisation for specific devices. The customisation can be applied broadly or on a very detailed level.

2) This approach is both flexible and extensible; filters can be modified and added at any time if needed.

Disadvantages

1) There is no standard way to manage similarities between mobile devices.

2) As more filters and device-specific functions are added to the application, the code can become verbose.

5.2.2.3 Development guideline – How to handle device diversity?

When developing an application for a specific mobile device it is easy to design functional modules and user interfaces according to the characteristics of the device. The results are usually satisfactory, but when targeting a range of different devices tradeoffs must be made. Developers can either use the J2ME approach to design applications that have only general functionalities that suit all requirements for each of the targeted devices, or adopt the .NET approach and write more device-specific code for each device to implement device-specific functions. The choice depends on the expectation of the specific mobile application.

5.3 Mobile application analysis

5.3.1 Mobile Web solution data processing model

Although both provide mobile Web application solutions, J2ME and .NET Mobile present two distinct mobile application development models. J2ME is used to build general-purpose applications on mobile devices of which Web applications are a part, while mobile ASP.NET is used solely to create Web-based applications. J2ME mobile Web applications (MIDlets) are strictly applications that run on mobile devices. .NET mobile Web applications are Web pages that are displayed by the client-side browsers.

5.3.1.1 J2ME model

The client side of J2ME mobile Web applications – the MIDlet - is a "smart" client approach, as well as a distributed computing model.

Advantages:

1) Being a distributed model, every mobile device shares the computing workload, which takes full advantage of growing computing power on mobile devices.

2) Only the necessary requests are sent to the Web Server. This significantly reduces network traffic and the request-response round trip.

3) Besides traditional Web controls, J2ME mobile applications are capable of processing more complicated client-side controls, which enhances application usability.

Disadvantages:

1) J2ME/MIDP runtime environment is required to be deployed on every mobile device.

2) Applications need to be downloaded to the device from the Internet, which can increase installation time.

3) The application consumes more client-side resources, memory, and local storage.

5.3.1.2 .NET model

.NET Mobile Solution is considered to be a lightweight client approach and at the same time a fully centralized model.

Advantages:

1) No additional client-side runtime deployment is required.

2) There is no download and installation cycle for .NET mobile Web Application; therefore, the application can be used in an immediate fashion.

3) Because data and information are handled on the server side, there is no critical requirement for the computing power of the mobile device. Even low-end devices can function properly.

4) As a centralised model, authentication and authorization are easier to controlled.

Disadvantages:

1) Being a fully centralised model, .NET Web servers are under great pressure. Every single user action is transmitted to the Web server, processed and sent back. This increases the server round trip dramatically. In a slow network environment it could become painful for the user if every action were to take a few seconds.

2) Only limited Web controls are used, therefore applications have less interactive features.

5.3.1.3 Development guideline – Which model to choose?

Knowing the differences between these two models, the question is which model should be adopted when developing mobile Web applications?

1) Using J2ME Model:

Applications that require strong client processing power, rich presentation, frequent and rapid user interaction, and do not require frequent communication with Web servers, are well suited for the J2ME model. Online mobile games is one such example.

2) Using .NET Mobile Model:

Applications that require frequent communication with Web servers, strong session tracking, and do not require client-side processing or frequent and rapid user interaction are suitable for the .NET model. For example, M-Commerce[22] applications.

5.3.1.4 Experiences & lessons

There are two mobile Web applications built for this project, the *"Rhodes University Mobile Meal Booking System"* and the *"Rhodes University Mobile Geographic Information System"*. The reader might have already noticed that the *"Mobile Meal Booking System"*, which is built using J2ME, would actually be more suited to development under the .NET Mobile environment, while the *"RU Mobile GIS"*, which is a .NET Mobile Web application, seems to fit the J2ME category more closely.

There are prices to pay for modelling in this way. When developing the *"Mobile Meal Booking System"*, most of the time was spent on setting up HTTP connection streams, handling incoming responses and writing control validation methods. In a similar vein, the constant refreshing of the screen after every single user action makes

[22] M-Commerce stands for Mobile Commerce. It is the E-Commerce implementation on mobile platforms.

the presentation of the *"RU Mobile GIS"* very uncomfortable.

The selection of platforms in each of the applications developed for this study was done fairly randomly. The results of this choice proved that what can be done under the J2ME development platform can also be done in a .NET environment, and vice versa. Furthermore, this selection of development environments emphasised that choosing the right model and tools is critical for application quality and Rapid Application Development.

5.3.2 Mobile Web application portability approach

As has been emphasised, one of the most critical factors that determines the success of a mobile Web application is its portability. A mobile application with high portability is able to run on top of a wide array of devices. To achieve application portability J2ME and .NET Mobile use two very different approaches.

5.3.2.1 J2ME approach:

J2ME mobile solution restricts the creation of mobile applications to one language – Java. In addition, all mobile devices that support J2ME applications are required to deploy the Java runtime environment and thereby support all functions provided by Java mobile Web applications. Any current mobile application can run on new devices if they have J2ME support.

Advantages:

Applications receive automatic support for devices, and can run on any new Java-enabled device directly without any modification.

Disadvantages:

Applications can only run on top of Java-enabled devices.

5.3.2.2 .NET approach:

The .NET mobile solution differs from the J2ME solution, it does not require client-side deployment. On the contrary it achieves portability by supporting all client-side implementations within the server-side application. This means that applications cannot be accessed directly by new mobile devices with a different standard, unless they are recompiled and redeployed within an updated development tool, which has added the support of the new standard.

Advantages:

Applications can run on top of virtually all devices and no client-side deployment is required.

Disadvantages:

Adding support to new devices and new standards can be time consuming and troublesome. Development tools and deployment environment modifications are involved.

5.3.2.3 Development guideline – Which approach to choose?

As a general rule: if an application is designed to target Java-enabled mobile devices, or the mobile Web application is developed and deployed in a heterogeneous environment, the J2ME approach should be implemented to achieve application portability. Otherwise the .NET portability approach can be used.

5.4 Graphical user interface designing

5.4.1 GUI rendering approach

5.4.1.1 J2ME approach:

J2ME uses the concepts of abstraction and discovery to render GUIs on mobile devices. Abstraction is used to specify GUIs conceptually, with the actual presentation being left to the MIDP implementation. The discovery concept allows the MIDlet to learn about the mobile device capability at runtime, rendering the GUI accordingly and programmatically. J2ME uses control objects defined inside the "javax.microedition.lcdui" package to construct GUIs. These controls are initialised by Java Virtual Machine, running on top of the mobile device.

5.4.1.2 .NET approach:

The .NET GUI rendering concept is much the same as that of J2ME. It uses an abstracted programming model for the development of mobile applications, which serves to shield developers from the differences between devices. The GUI presentation of these controls depend on the client browser. .NET Mobile uses XML to compose mobile controls, and translates XML into other markup formats such as WML and cHTML accordingly upon receiving requests from specific mobile devices.

5.4.1.3 Development guideline – How to render GUIs for multiple devices?

In order to design and develop GUIs for a wide range of mobile devices the concept of abstraction must be adopted. Only abstracted GUI components guarantee the correct delivery of functions. Presentations vary from one device to another, so developers should rather design GUI components for their functionality than presentational quality.

5.4.2 GUI design approach

J2ME and .NET share a similar mechanism to organize the display of on-screen GUI components. J2ME uses the "Screen" object while .NET uses the "MobilePage" object to present a screen, which is the root of all other GUI components and containers. Both use the "Form" object as the top-level container within a screen to organize and contain other GUI controls.

5.4.2.1 On screen display design

The design of the graphical user interface can be divided into three different levels, namely application level, page level and form level.

Application level design (What to present)

The application level is a conceptual level. In this level designers should define what items of information to present, then classify them and pick only the essential ones. They should then decide what kind of user controls to use to present the information, and use the same types of controls to present the same types of information.

Page level design (Where to present)

At the page design level information is further categorized into several modules, such as data input modules, error message modules and user confirmation modules. The number of "Forms" used to contain these modules should be assigned and the display sequence of the forms should be decided.

Form level design (How to present)

Forms are actually GUI components that are displayed on the screens of mobile devices. All other GUI components, such as buttons and text boxes, reside inside the form. At this level designers should focus on the display layout or sequence of those GUI components.

5.4.2.2 Development guideline – How to design GUIs with high usability

Only present necessary information to the user and minimize user interaction. Maintain consistency when using GUI controls. Separate large content into several modules. Minimize scrolling within a page, rather using another form if necessary. Restrict the GUI components within a form and provide alternative methods for accessing the same information.

5.4.3 GUI navigation design

Besides presenting on-screen information for the users, another very important task of GUI control is navigation. J2ME uses "Command" and "List" objects while .NET provides "Command", "List" and "Link" controls to navigate through forms.

5.4.3.1 Using system built-in buttons for navigation

The system built-in back button provides a convenient way for users to navigate through forms. Direction buttons are used to browse items within a form while the back button is used to return to the previous form. When dealing with session states, the built-in back button can be problematic. Usually, in mobile Web applications, the state of an application corresponds to the display of the user interface. The problem arises if the user clicks on the back button, since this brings up a page generated from old and invalid states, and can often result in data inconsistencies.

5.4.3.2 Using soft buttons for navigation

A soft button is one that has no predefined function and label, thus allowing developers to assign a value programmatically and render the display during runtime.

Soft buttons are excellent for navigation; they can be programmed to navigate to any location within the application. Soft buttons can be used for proper backward navigation, thus avoiding the aforementioned problem.

5.4.3.3 Development guideline – How to arrange the navigation?

Always provide buttons for the user to return to their previous step and, specifically to the initial level of the application. Use short, clear, meaningful labels to inform users of their location. Divide the list into multiple groups and display them in different forms when there are too many items in the list. Never define more than two soft buttons and beware of the behavior of the system built-in back button.

5.5 Development environment and tools

5.5.1 J2ME development environment evaluation

Language and platform support

Java is the only language that can be used to build J2ME mobile Web applications, but the development environment is available on all major platforms, such as Linux, Windows and Solaris.

Development environment installation and configuration

Although installing and configuring the J2ME mobile application development environment is a fairly complicated process the whole environment is small in size, free for everyone, open source and available on all major platforms [see section 3.2.1].

Graphical user interface design

Although SUN provides an integrated development environment – the SunONE Studio Mobile Edition - for the development of J2ME mobile applications, the lack of graphical tools for designing GUIs makes it less useful, as well as the fact that all controls have to be written from scratch.

Client and Server-side development

Client-side and Server-side developments are completely separate processes and performed under entirely different development environments. Programming of communication modules, HTTP connections and database connections has to be done manually [Please refer to section 3.4.3.2 and 3.4.4.1].

Deployment

J2ME Wireless toolkit gives developers a graphical tool to package and deploy mobile Web applications. The deployment of a J2ME mobile Web application has two parts, the server-side deployment and the client-side deployment. The server-side application is deployed on a Web server using the J2EE Deployment Tool while the client-side application, a JRE file, is downloaded through the Internet and run on the mobile device [Please see section 3.3.5.4 and section 3.4.5.2].

Mobile application testing utilities

As mentioned in section 3.3.5.3, a range of mobile device emulators are integrated into the J2ME Wireless Toolkits that can be used to test J2ME mobile Web applications. J2ME Wireless Toolkits also provide an excellent set of testing and debugging facilities for application monitoring and examination.

Help and API system

Three HTML format APIs – the J2SE API, the J2ME API and the J2EE API are used. Code examples can be downloaded as separate packages.

5.5.2 .NET Mobile development environment evaluation

The development of mobile Web application under the .NET environment takes advantage of the full support of Visual Studio.NET integrated development environment. Visual Studio.NET is the latest development tool developed by Microsoft. It reduces the complexity of mobile Web applications development. Using Visual Studio .NET, allows developers to build applications that target the Web and virtually any mobile device.

Language and platform support

A .NET mobile Web application can be written in virtually any language, C#, VB.NET, JScript and so forth. Development can only be carried out under Microsoft Windows platform however.

Development environment installation and configuration

As was introduced in section 4.2.1, all the development tools that are needed to create .NET mobile Web applications are provided in the "install wizard". Simply follow the instructions to complete installation with no manual configuration required. Currently the .NET mobile application development environment is only available on Microsoft Windows Platforms.

Graphical user interface design

Visual Studio.NET is a fully graphical development environment: a graphical designer is provided to design and develop GUIs and most of the GUI codes are generated automatically [Please refer to section 4.4.2.2].

Client and Server-side design

The client side of .NET mobile applications is a Web browser; it can be a simple WML browser or a full-featured HTML browser. On the server side, database connections can be set up by using the Server browser [see section 4.5.1.4] with no manual setting up of HTTP connections required - the browser takes care of this when sending requests to the Web server.

Mobile application testing utilities

Visual Studio.NET offers an HTML-based testing and debugging tool for developers to monitor the states and parameters of mobile Web applications. No mobile device emulator is supplied by Visual Studio.NET. For device simulating external software has to be used [Please see section 4.4.2.3].

Deployment

Visual Studio.NET gives developers a set of tools to create a mobile Web application installer that can easily run on a target deployment system. .NET mobile Web applications, on the other hand, can only be deployed on Microsoft Windows platforms.

Help and API system

As was discussed in section 4.4.2.2, Visual Studio.NET provides an excellent help and API system, which dynamically retrieves and displays information on a desired topic by simple highlighting the keyword.

5.5.3 Development guideline – What is an ideal environment?

To enable developers to build mobile Web applications with high performance, good usability and strong reliability, rapidly and efficiently, a development tool should meet the following requirements.

1) Object Oriented programming and modular design

The development environment should support object oriented programming and modular design. This entails a clear separation of application business logic and application user interface presentation. When the application logic and presentation layer are separated, multiple devices can have a different "look-and-feel" while sharing the same logic module. This reduces the total amount of code and increases the code reusability.

2) Rich utilities

The development environment should have rich utilities that cover the design, coding, testing and deployment of an application.

3) Fully graphical environment for GUI design

The development environment should have graphical "GUI component designers", thus greatly improving the development efficiency. User interfaces can be placed in applications using a drag-and-drop method, automatically generating codes. In this way developers can concentrate more on the development of application logic than on the user interface.

3) Multiple mobile devices support

Applications built using the development environment should be able to run on a large variety of mobile devices – from low-end devices such as WML enabled cell phones to high-end devices such as the HTML based Pocket PCs.

4) Customisability

The development environment should be customisable, which means the environment should be able to be configured to suit different development requirements.

5) Extensibility

The development environment should be extensible, enabling new devices to be supported in the future. This ensures that mobile Web applications developed today can run on any new mobile devices yet to come.

6) Portability

Applications (server-side) built using the development environment should be able to run on all major platforms.

7) Easy-to-use help and API system

Help systems and APIs are considered to be one of the most frequently used utilities in development applications. Undoubtedly, an easy and efficient help and API system saves on development time.

5.6 Chapter review

In this chapter, firstly, comparisons between J2ME and .NET mobile Web application development and deployment were made, also between the applications themselves. Secondly, guidelines were given on how to deal with general limitations and the diversity of mobile devices using both J2ME and .NET mobile solutions. We then discussed mobile Web Solution Data Processing Models and portability approaches. After that we outlined guidelines of Graphical User Interface design, including a GUI rendering approach, GUI design approach and GUI navigation design method. Finally, we drew a general guideline of an ideal development environment for mobile Web applications. By following these guidelines the developer is able to develop highly portable and extensible mobile Web applications rapidly and efficiently, with high performance, good usability and strong reliability.

Chapter 6: Concluding Comments

– *"It is never finished. There is always the next objective, the next goal"* [Quotable Quotes].

This chapter reviews the research work that has been accomplished for this project, draws conclusions based on the experience and results gained during the course of the research, lists goals achieved, and suggests possibilities for future work arising out of this study.

6.1 Review

This thesis researches the development and deployment life cycle of the Java mobile Web application and the .NET mobile Web application environments, through experience gained from the development of mobile applications using each of these technologies. The study is aimed at providing a developer with an evaluation of how the technologies stand up to each other, and a set of guidelines on how to efficiently create high-quality mobile Web applications.

In chapter one, a background introduction of this project was presented, defining the goals to be achieved. In chapter two we introduced and compared a choice of mobile communication mechanisms, which included paging communication, cellular communication, and mobile data communication. We also gave an architectural overview of the current mobile Web protocols – WAP, imode, J2ME and .NET. In chapter three we explored the development life cycle of J2ME mobile Web applications by examining the development details of the *"Rhodes University Mobile Meal booking System*". The discussion covered mobile application design, coding, testing and deployment. In chapter four we investigated the development life cycle of the .NET mobile Web application by examining the development detail of the *"Rhodes University Mobile Geographic Information System"*. Finally in chapter five, we conducted a comparative analysis of the development of J2ME and .NET mobile Web applications, delivering guidelines on the development and deployment of such applications.

6.2 Related work

Because of the nature of experimental and comparative system building, the mobile Web applications described in this report were built with two entirely different development environments on two disparate platforms, Linux and Windows.

Inevitably, a large variety of technologies were involved. These are described here for the information of readers intending to repeat or extend any aspect of this investigation.

1) C#, a new programming language:

Since the C# is a fairly new language that has recently been dispatched with .NET by Microsoft, some effort was invested in gaining proficiency in this language.

2) APIs

Although the J2ME mobile Web applications were created in Java, J2ME supplies a completely different API with new classes, methods and fields. The same situation also occurs with the J2EE API, since J2EE is used to develop server-side components for the *"Mobile Meal Booking System"*. The .NET framework also provides a huge collection of class libraries to be explored.

3) A range of IDEs and Toolkits

There are a large number of tools and IDEs used for the creation of the Java mobile Web application and .NET mobile Web application. These include Java 2 SDK mobile edition, standard edition and enterprise edition, J2ME Wireless Toolkit, Microsoft Mobile Internet Toolkit, SunONE Studio Mobile Edition and Microsoft Visual Studio.NET.

4) Configuration of Database management systems

In a Linux environment, the MySQL database management system only provides shell-prompt-based utilities; a large number of commands need to be remembered for database operation and administration.

5) Configuration of various mobile device emulators

To test mobile Web applications, a wide range of emulators was used. To get the most convincing results that are closest to real conditions, various settings were required to be configured properly.

6) Writing demos

To become familiar and comfortable with the development environment and to get a better understanding of the chosen languages, it is necessary to experiment with tens, if not hundreds, of smaller demo applications. These should be written before large applications are attempted.

7) Testing applications on real mobile devices

Besides the tests conducted using mobile device emulators, mobile Web applications that were created for this project were also tested on real mobile devices such as the Pocket PC.

6.3 Overall achievement

The ultimate goal of this research is to provide a comparative analysis of the two chosen development environments, and a set of guidelines for the efficient and reliable development and deployment of mobile Web applications, with a focus upon application portability and extensibility.

1) Investigations were made into the current state of technologies involved in the creation of applications in the mobile telecommunications and mobile web arena.

2) Two fully functional mobile web applications were created – the "Rhodes University Mobile Meal Booking System" and the "Rhodes University Mobile Geographical Information System".

3) The development and deployment life cycle of both J2ME and the .NET mobile Web application were investigated in detail, and compared with one other.

4) The differences between J2ME and the .NET mobile Web application development environments were cataloged, and classes of problems were identified where each might be preferable.

5) Software Engineering guidelines were drawn up for the development and deployment of mobile Web applications.

6.4. Evaluation of achievement

6.4.1 Platform comparison

In this project, detailed comparisons of J2ME and .NET mobile approaches are based on two fully functional pilot mobile systems, the *"Rhodes University Mobile Meal Booking System"* and the *"Rhodes University Mobile GIS"*. The comparisons between J2ME and .NET mobile approaches are comprehensive. They cover not only the mobile Web applications themselves, but also the entire development and deployment life cycle. The integrated development environment, the application SDK, the coding environment, the graphical user interface design environment and help systems of both J2ME and .NET mobile solutions are compared. Comparisons of the J2ME and the .NET mobile approaches include not only those differences on mobile devices, but also on the server side. On the mobile client side, runtime requirements, local storage requirements, client-side data processing abilities and event handling models are compared. On the server side, aspects of web server support and server-side technology support are compared. General comparisons of platform dependency, language support, application type and executing speed are also made. In all comparisons, we emphasized the compression of different computational models used by J2ME and .NET mobile Web applications, how those models are applied to mobile Web application design and deployment, and how they impact the overall performance of mobile Web applications. We highlighted the comparison of development tools and environments of J2ME and .NET mobile solutions. We also addressed GUI design issues while analyzing J2ME and .NET mobile Web application development environments.

6.4.2 Guideline delivery

After recapitulative and comprehensive comparisons were made, general guidelines were drawn based on the observations and analysis of both J2ME and .NET mobile approaches. These guidelines include how to handle mobile device limitations, how to deal with device diversity, how to choose a mobile computational model, how to choose mobile application portability approaches, how to render GUIs for multiple devices, how to design GUIs with high usability, how to arrange the navigation and what an ideal Environment consists of. These guidelines cover issues ranging from the choice of mobile application fundamental computation model to the use of design tools usage. They therefore give considered advise on how to develop highly portable and extensible mobile Web applications with high performance, good usability and strong reliability.

6.4.3 Limitations of the study

Although the comparison and evaluation of the J2ME and .NET mobile Web application approaches made in this thesis cover a wide range of aspects and deal with real working development and deployment conditions, there are several limitations to the study.

1) Two different pilot implementations were used for the two different technologies. This was a deliberate research decision to ensure that the design as well as the implementation phases of the two different philosopies could be measured in terms of development time and functional underpinning. The thinking was that, if a single pilot implementation was used, the second environment in which it was implemented would be advantaged due to the greater understanding of the problem domain gained during its implementation in the

first environment. This could of course have been overcome by using a different designer and implementor for each of the two environments under consideration, which would have introduced other forms of measurement uncertainty in terms of the rate of output of the different developers, and would have been outside of the scope of this project as an individual MSc thesis. Consequently, the best overall approach was considered to be the use of two equally challenging, but different problems, implemented by the same author in the two different environments The consequence of this trade-off was that direct comparisons of specific aspects of the systems could only be done in general terms, and not as a specific comparison such as one might be able to do if the same application were given to two monitored development teams working on an identical problem, but in the two different development environments.

2) User group testing is the best way to test, measure, compare and evaluate the efficiency, performance usability and reliability of two applications. There are no such user control groups available in the limited scope of this MSc project to get these objective measures of usability and reliability. These impressions were taken from a limited group of people who have experimented with the two pilot systems.

3) Due to the limited funding for this project, most of the tests of both J2ME and .NET mobile Web applications were conducted using software emulators. Software emulators are good for simulating the functionality of mobile Web applications, but are not sufficiently accurate for performance measuring since desktop computers are far more powerful than most existing mobile devices.

4) For both J2ME and the .NET mobile Web applications, HTTP communication is established under a Wired and Wireless Local Area Network. Because of a lack of compatible mobile devices and mobile Internet service coverage, we were not able to observe application behaviours under a real wireless telecommunication network environment such as GSM or CDMA.

6.5 Possible future extensions

6.5.1 An investigation into Mobile Web application security

Security issues always demand much attention and are often the source of debates about the adoption of wireless communications and mobile Web applications. They fell beyond the scope of this investigation, but would constitute a useful extension to the project.

6.5.2. Mobile Web application and XML Web Services integration

"XML Web services are programs residing either locally on a PC or remotely on a server. These services communicate via the standard protocols HTTP and SOAP, and the methods of XML Web services can be accessed in a location-independent manner" [Wigley, Andy]. This is an ideal programming model for a highly distributed system such as the mobile Web application. With the integration of XML Web Services, Mobile Web applications could access a greater range of functionality than with the more limited mobile controls that this investigation was restricted to. This extension would also provide a promising architecture for dynamic application construction, and the creation of a new software business model, i.e. renting software over the Internet as needed.

6.6 Overall conclusions

In this thesis, objective comparisons and evaluations of both the J2ME and .NET mobile solutions are conducted. Their strong points are highlighted and their limitations are addressed. Guidelines are also drawn for rational choices for the design, creation and deployment of mobile Web applications using these two technologies.

The rivalry between Java and Microsoft .NET technologies in terms of brand loyalty and popular support has been debated in the trade press and in faculty tearooms for some time. Although the J2ME and .NET mobile Web application solutions are subcategories of each of these technologies, and are thereby competitors, this thesis does not constitute an extension of this debate into the mobile arena. Nor does it purport to judge whether a J2ME or .NET mobile solution is the overall better choice for all situations. J2ME and .NET mobile solutions use different models and approaches, and solve problems in different ways. Each has strong points in specific problem domains or scenarios, and weaknesses in others, but neither is an overwhelming winner.

At present, mobile technologies are on a rapid innovation and development curve. Old and inadaptable technologies are quickly replaced by more efficient and flexible alternatives. J2ME and .NET's differences give them competitive strengths in different areas of mobile development, so neither is easily cast aside, and both currently contribute greatly to the expansion, popularization, and evolution of wireless communication and of mobile Web application technologies. They are likely to co-exist with each other, and possibly other emerging mobile technologies, for the foreseeable future.

References

[Aldridge, Irene] *Analysis of Existing Wireless Communication Protocols* By Irene Aldridge, Columbia University New York, NY, USA, at: http://www.columbia.edu/~ir94/wireless.html

[Apache] Apache Official Website: http://httpd.apache.org

[Bak, Lars] *The Need for Speed in Mobile Devices* by Lars Bak; Sun Microsystems, Inc.

[Butler, M. et al] *ASP.NET Mobile Controls: Tutorial Guide: Adaptive Web Content for Mobile Devices with the MMIT*; by Matt Butler, Matt Gibbs, Costas Hadjisotiriou and so on; ISBN: 1861005229; Publisher: Wrox Press Inc

[Choksi, Dipal] Testing Mobile application with Device Emulators by Dipal Choksi at http://www.dotnetforce.com

[D'Ambra, P. et al] *Advanced environments for parallel and distributed applications: a view of current status*; by Pasqua D'Ambra, Marco Danelutto, Daniela di Serafino and Marco Lapegna; "Parallel Computing" volume 28, issue 12, 2002

[Farley, Jim] *Microsoft .NET vs. J2EE: How Do They Stack Up?* by Jim Farley, at: http://java.oreilly.com/news/farley_0800.html

[Ferguson, Derek] *Mobile .NET*, by Derek Ferguson, APress press, ISBN: 1-89311571-2

[Garvin, John] Small-Footprint Implementations of the Java Virtual Machine; by John Garvin; March 22, 2000

[GATs' Guide] Found on *The GAT's Guide to Website Construciton* at http://www-english.tamu.edu/pers/grad/coste/how2/step3.html

[Governor, James] *In praise of the .Net vision*, James Governor, Computing, 2002

[Gruhn, V. et al] *Software processes for the development of electronic commerce systems*, by Volker Gruhn and Lothar Schöpe, "Information and Software Technology" volume 44 issue 14; 2002

[Hadjisotiriou, Costas] *Web programming for Mobile Devices using ASP.NET Mobile Controls*, by Costas Hadjisotiriou, ISBN: B00006L8GZ, Publisher: Wrox

[Harkey, D. et al] *Wireless Java Programming,* by Dan Harkey, Shan Appajodu, Mike Larkin, ISBN: 0-471-21878-2, Publisher: Wiley Publishing, Inc.

[Hewlett-Packard] *Communications Industry Overview*, Hewlett-Packard, 2002

[IBM] *Wireless: Increasing and extending the benefits of information technology beyond wired networks*, IBM Corporation, 2001

[LEE, Raymond Shu-Tak] *J2ME Tutorial*, LEE, Raymond Shu-Tak, at: http://www4.comp.polyu.edu.hk/~csstlee/csstlee.html

[LEE, Raymond Shu-Tak] *Servlet Tutorial*, LEE, Raymond Shu-Tak, at: http://www4.comp.polyu.edu.hk/~csstlee/csstlee.html

[Leiner, Barry] *A Brief History of the Internet* by Barry M. Leiner et al at: http://www.isoc.org/internet

[Leopold, Eileen] *International Presentation SA*, Eileen Leopold and Associates, 14 May 2001

[Leung, Linda] *Java versus .Net debate heats up*; Linda Leung, Comdex, Fall 2000, Las Vegas.

[Lewis, John] *Java software solutions*, by John Lewis and William Loftus, Addison-Wesley Press, ISBN: 0-201-72597-5

[Madria, S. et al] *Mobile data and transaction management*; by Sanjay Kumar Madria, Mukesh Mohania, Sourav S. Bhowmick and Bharat Bhargava; "Information Sciences"; volume 141; issue 3-4; 2002

[Mandrake] Mandrake Official Website, at: www.mandrakeuser.org/docs/mdoc/ref/about.html

[Meredith, Simon] *Cutting a dash with .NET*, Simon Meredith, Computer Reseller, 18 Sept 2000.

[Microsoft] *ADO.NET Architecture*, MSDN, at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconadonetarchitecture.asp

[Microsoft] *Defining the Basic Elements of .NET* http://www.microsoft.com/net/basics/whatis.asp

[Microsoft] Microsoft Official Web Site, at:
www.microsoft.com/windowsxp/pro/howtobuy/pricingretail.asp

[Miller, Jim] *The .NET Vision*; Jim Miller, Presentation on Microsoft .NET Crash Course for Faculty and PhDs, Cambridge, 3 Sep 2001.

[Milroy, S. et al] *.NET Mobile Web Developer's Guide*; by Steve Milroy, Ken Cox and so on; ISBN: B00006929W; Publisher: Syngress Publishing

[Muchow, John] Core J2ME Technology & MIDP, by John W. Muchow, ISBN: 0-13-066911-3, Sun Microsystems Press

[MySQL] MySQL Official Website, at:
http://www.mysql.com/products/mysql/index.html

[Nadel, Brian] *Waiting For the Wireless Revolution*, By Brian Nadel, May 21, 2002, at: http://www.pcmag.com/article2/0,4149,59644,00.asp

[Netcraft] *Netcraft Web Server Survey*, at: http://www.Netcraft.com/survey

[NTT DoCoMo] NTT DoCoMo Official English Web Site, at:
http://www.nttdocomo.co.jp/english

[Olive, David] *Chairman's Public Policy Report 2002*, David A. Olive, WITSA Public Policy Working Group Australia, 26 Feb 2002.

[Poulton, Austin] *Enterprise Java Notes*, 2002 Rhodes University Computer Science Honours Course, by Austin Poulton

[Price, David] *Developing MIDP Client/Server Applications*; by David Price and James Reilly; Sun's 2001 Worldwide Java Developer Conference

[PricewaterhouseCoopers] *Technology Forecast: 2002-2004 Volume 1*, PricewaterhouseCoopers, ISBN: 1-891865-05-6

[PricewaterhouseCoopers] *Technology Forecast: 2002-2004 Volume 2*, PricewaterhouseCoopers, ISBN: 1-891865-06-4

[Quotable Quotes] Found on *The Quotable Quotes*, at:
http://ics.usm.my/quotes.asp

[Ridgeway, Mark] *.NET Wireless Programming*; by Mark Ridgeway; ISBN: 0782129757; Publisher: Sybex

[Sun Microsystems] *Developing Wireless Applications using the Java 2 Platform, Micro Edition Bill Day Technology Evangelist* Sun Microsystems, Inc. at: http://www.billday.com

[Sun Microsystems] *J2ME Archive*, at: http://billday.com/j2me/

[Sun Microsystems] *J2ME Enabled Devices*, at http://wireless.java.sun.com/device

[Sun Microsystems] *J2ME FAQ*, at java.sun.com/j2me/faq.html

[Sun Microsystems] *J2EE vs. .NET*, white paper from SUN, SUN, 2001

[Sun Microsystems] *The J2EE Tutorial*, at:
http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html

[Taylor, Michael] *J2ME IDE Comparison prepared*; by Michael Taylor; 29 June 2002 Version 1.1; 2002 Developnet Consulting Limited.

[Un progetto di] *Java vs. C#*, project by Un progetto di, at:
http://www.java-vs-csharp.com/javavscsharp/Home.en.xml

[W3C] *W3C online tutorial*, at:
http://www.w3schools.com/

[Wigley, Andy] *Building .NET Applications for Mobile Devices*, Andy Wigley, Peter Roxburgh, Microsoft Press, ISBN: 0-7356-1532-2, 2002

[Zhao, XG. et al] *Evaluate .NET for Mobile solutions* Southern Africa Telecommunication and Networking Conference (SATANC) 2002 by Xiaogeng Zhao, Peter Clayton

Appendix

Appendix A – J2ME Emulators SDKs, and IDEs

A Complete List of J2ME Emulators/Simulators, SDKs, and IDEs			
Tool	**Contact**	**Brief description**	**Compatibility**
J2ME Wireless Toolkit 18 Apr 2002	Sun Microsystems	Develop J2ME MIDP *midlets* on Solaris, Linux, and Win32. Optionally plugs into Sun's free Forte for Java Community Edition and Borland jBuilder MobileSet.	CLDC 1.0, MIDP 1.0
J2ME MIDP 18 Apr 2002	Sun Microsystems	J2ME Mobile Information Device Profile reference implementation. MIDP for PalmOS also available.	CLDC 1.0, MIDP 1.0
MIDP for PalmOS 18 Apr 2002	Sun Microsystems	MIDP or PalmOS devices. Develop using PalmOS Emulator (see below).	CLDC 1.0, MIDP 1.0
PalmOS Emulator 18 Apr 2002	Palm, Inc.	The PalmOS Emulator, for use with Sun's PalmOS version of the MIDP. Versions for Win32, Unix, MacOS developers.	CLDC 1.0, MIDP 1.0
Nokia J2ME Tools 18 Apr 2002	Nokia	Target Nokia J2ME devices using these tools. Nokia Developer's Suite for J2ME plugs into Forte for Java and Borland jBuilder MobileSet.	CLDC 1.0, MIDP 1.0
Siemens Mobility Toolkit 18 Apr 2002	Siemens	Develop J2ME applications targeted at Siemen's	CLDC 1.0, MIDP 1.0

		J2ME-enabled GSM phones.	
RIM Blackberry JDE 18 Apr 2002	Research in Motion	Full IDE and SDK for developing J2ME applications targeted at RIM's wireless handhelds. Runs on Win32. (RIM Blackberry skin also bundled with J2ME Wireless Toolkit)	CLDC 1.0, MIDP 1.0
Motorola iDEN Tools 18 Apr 2002	Motorola	Develop J2ME applications targeted at Motorola's J2ME-enabled iDEN mobile phones.	CLDC 1.0, MIDP 1.0
ME4SE 25 Apr 2002	Stefan Haustein	Provides support for J2ME APIs (LCDUI, Generic Connection Framework, etc.) on the J2SE Platform and PersonalJava devices.	CLDC 1.0, MIDP 1.0
WHITEboard Wireless Java SDK 18 Apr 2002	Zucotto Wireless	Full SDK and IDE for developers building wireless J2ME apps and services.	CLDC 1.0, MIDP 1.0
MicroEmulator 18 Apr 2002	Bartek Teodorczyk	Applet based J2ME device emulator.	CLDC 1.0, MIDP 1.0
Yospace Motorola Accompli 008 Emulator 18 Apr 2002	Yospace	An online J2ME emulator. You can run the provided midlets or upload your own.	CLDC 1.0, MIDP 1.0
Jbed Micro Edition 18 Apr 2002	esmertec	J2ME CLDC and MIDP implementation for mobile devices.	CLDC 1.0, MIDP 1.0
Forte for Java 18 Apr 2002	Sun	Develop J2ME apps using Forte for Java. FFJ 3.0 Community Edition is free, and the J2ME Wireless Toolkit, Siemens Mobility Toolkit, and	CLDC 1.0, MIDP 1.0

		Nokia Developer Suite for J2ME all integrate into it.	
jBuilder MobileSet 18 Apr 2002	Borland	Develop J2ME apps using the jBuilder IDE. Special Nokia Edition for targeting Nokia J2ME devices.	CLDC 1.0, MIDP 1.0
J2ME CLDC 18 Apr 2002	Sun Microsystems	The J2ME CLDC implemented using the K Virtual Machine. SDK and tools for Solaris, Linux, and Win32 development.	CLDC 1.0
xKVM (previously Color KVM) 18 Apr 2002	Stefan Haustein and Michael Kroll	Port of Sun's CLDC which implements 8-bit color support for capable PalmOS based devices. Works with Emulator	CLDC 1.0
iEmulator 18 Apr 2002	Taisuke Fukuno	Another emulator help you build *iAppli* applications for NTT DoCoMo's imode Java service.	CLDC 1.0, iAppli specs
iJADE Lite 18 Apr 2002	Zentek Technology	Develop imode Java applications using this emulator for various NTT DoCoMo 503i *iAppli* mobile phones.	CLDC 1.0, iAppli specs
LG TeleCom ez-Java emulator 18 Apr 2002	LG TeleCom online support (in Korean)	Develop applications for LG Telecom's J2ME-based *ez-Java* mobile phone service.	CLDC
J2ME CDC 18 Apr 2002	Sun Microsystems	The J2ME CDC using the CVM for Linux	CDC 1.0
J2ME Foundation 18 Apr 2002	Sun Microsystems	Built on the CDC, for Linux and VxWorks.	CDC 1.0

Source: [Sun Microsystems]

Appendix B – Mobile Device Emulators For MMIT

A List of Mobile Device Emulators for MMIT		
Device Emulator	**Browser and Version**	**Support added in**
Jungle Emulator	i-page master 1.0 (502i)	MMIT 1.0
Ericsson R380 Emulator	Ericsson R380 2.0	MMIT 1.0
Microsoft Emulator	Microsoft Mobile Explorer 2.01	MMIT 1.0
Microsoft Emulator	Microsoft Mobile Explorer 3.0 (HTML)	MMIT 1.0
Microsoft Emulator	Microsoft Pocket Internet Explorer 2000 (4.01)	MMIT 1.0
Nokia Mobile Internet Toolkit 3.0 with default skin	Nokia Mobile Browser 3.0	Device Update 1
Nokia Mobile Internet Toolkit 3.0 with 3330/3395 skins	Nokia Mobile Browser 3.05	Device Update 1
Open Wave SDK	Open Wave 6.0	-

Source: [Choksi, Dipal]

Appendix C – J2ME Enabled Mobile Devices

A Complete List of J2ME Enabled Mobile Devices					
Casio	A3012CA	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	132x176/14 bits
Casio	C452CA	CDMA	800	MIDP 1.0, CLDC 1.0	120x133/8 bits
Fujitsu	F503i	PDC	800	CLDC 1.0	120x130/8 bits
Fujitsu	F503iS	PDC	800	CLDC 1.0	120x130/10 bits
Fujitsu	F504i	PDC	800	CLDC 1.0	16 bits
Hitachi	C3001H	CDMA	800	MIDP 1.0, CLDC 1.0	120x162/12 bits
Hitachi	C451H	CDMA	800	MIDP 1.0, CLDC 1.0	120x143/8 bits
Kyocera	C3002K	CDMA	800	MIDP 1.0, CLDC 1.0	120x160/16 bits
LG Electronics	C-nain 2000	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	120x133/8 bits
LG Electronics	C-nain 2100	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	8 bits
LG Electronics	CX-300L	CDMA	1900	CLDC 1.0	120x160/8 bits
LG Electronics	Cyber-ez-X 1	CDMA	1900	CLDC 1.0	128x128/2 bits
LG Electronics	I-Book	CDMA	1900	CLDC 1.0	128x128/2 bits
LG InfoComm	LX5350	AMPS, CDMA2000 1X	800, 1900	MIDP 1.0, CLDC 1.0	120x198/16 bits
LG InfoComm	VX1	AMPS, CDMA2000 1X	800, 1900	MIDP 1.0, CLDC 1.0	128x104
Mitsubishi	D2101V	W-CDMA		CLDC 1.0	132x162/18 bits
Mitsubishi	D503i	PDC	800	CLDC 1.0	132x142/10 bits
Mitsubishi	D503iS	PDC	800	CLDC 1.0	132x142/10 bits
Mitsubishi	D504i	PDC	800	CLDC 1.0	18 bits
Mitsubishi	J-D05	PDC	1500	MIDP 1.0, CLDC 1.0	12 bits
Motorola	A388	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	2 bits

Motorola	A820	GSM/GPRS, W-CDMA	900, 1800, 1900		176x220/12 bits
Motorola	Accompli 008/6288	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	320x240/2 bits
Motorola	Accompli 009	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	240x160/8 bits
Motorola	i50sx	iDEN	800	MIDP 1.0, CLDC 1.0	111x100/2 bits
Motorola	i55sr	iDEN	800	MIDP 1.0, CLDC 1.0	111x100/2 bits
Motorola	i80s	iDEN	800	MIDP 1.0, CLDC 1.0	119x64/1 bit
Motorola	i85s	iDEN	800	MIDP 1.0, CLDC 1.0	111x100/2 bits
Motorola	i90c	iDEN	800	MIDP 1.0, CLDC 1.0	111x110/2 bits
Motorola	i95cl	iDEN	800	MIDP 1.0, CLDC 1.0	120x160/8 bits
Motorola	T280i	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	
Motorola	T720	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	120x160/12 bits
Motorola	T720	AMPS, CDMA2000 1X	800, 1900	MIDP 1.0, CLDC 1.0	120x160/12 bits
Motorola	V60i	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	96x64
Motorola	V60i	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0	96x64
Motorola	V60i	AMPS, TDMA	800, 1900	MIDP 1.0, CLDC 1.0	96x64
Motorola	V66i	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	96x64
NEC	N2002	W-CDMA		CLDC 1.0	16 bits
NEC	N503i	PDC	800	CLDC 1.0	120x130/10 bits
NEC	N503iS	PDC	800	CLDC 1.0	120x130/10 bits

NEC	N504i	PDC	800	CLDC 1.0	16 bits
Nokia	3410	GSM	900, 1800	MIDP 1.0, CLDC 1.0	96x65/1 bit
Nokia	3510i	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	96x65/12 bits
Nokia	3530	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	96x65/12 bits
Nokia	3570	CDMA2000 1X	1900	MIDP 1.0	96x65/2 bits
Nokia	3585	AMPS, CDMA2000 1X	800, 1900	MIDP 1.0, CLDC 1.0	96x65/2 bits
Nokia	3585i	AMPS, CDMA2000 1X	800, 1900	MIDP 1.0, CLDC 1.0	96x65/2 bits
Nokia	3590	GSM/GPRS	850, 1900	MIDP 1.0, CLDC 1.0	96x65/1 bit
Nokia	3650	GSM	900, 1800, 1900	WMA 1.0, MMAPI 1.0, MIDP 1.0, CLDC 1.0	176x208/12 bits
Nokia	5100	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	128x128/12 bits
Nokia	6100	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	128x128/12 bits
Nokia	6200	GSM/GPRS	850, 1800, 1900	MIDP 1.0, CLDC 1.0	128x128/12 bits
Nokia	6310i	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	95x65/1 bit
Nokia	6610	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	128x128/12 bits
Nokia	6650	GSM/GPRS, W-CDMA	900, 1800	MIDP 1.0, CLDC 1.0	128x160/12 bits
Nokia	6800	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	128x128/12 bits
Nokia	6800	GSM/GPRS	850, 1900	MIDP 1.0, CLDC 1.0	128x128/12 bits
Nokia	7210	GSM/GPRS	900, 1800,	MIDP 1.0, CLDC 1.0	128x128/12 bits

			1900		
Nokia	7250	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	128x128/12 bits
Nokia	7650	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	176x208/12 bits
Nokia	8910i	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	96x65/12 bits
Nokia	9210 Communicator	GSM	900, 1800	MIDP 1.0, CLDC 1.0, JavaPhone 1.0, PersonalJava 1.1.1	640x200/12 bits
Nokia	9210i Communicator	GSM	900, 1800	MIDP 1.0, CLDC 1.0, JavaPhone 1.0, PersonalJava 1.1.1	640x200/12 bits
Nokia	9290 Communicator	GSM	1900	MIDP 1.0, CLDC 1.0, JavaPhone 1.0, PersonalJava 1.1.1	640x200/12 bits
Panasonic	C3003P	CDMA	800	MIDP 1.0, CLDC 1.0	132x176/16 bits
Panasonic	P2101V	W-CDMA		CLDC 1.0	176x220/18 bits
Panasonic	P503i	PDC	800	CLDC 1.0	120x130/8 bits
Panasonic	P503iS	PDC	800	CLDC 1.0	120x130/8 bits
Panasonic	P504i	PDC	800	CLDC 1.0	16 bits
Research In Motion	Blackberry 5810	GSM/GPRS	1900	MIDP 1.0, CLDC 1.0	160x160/1 bit
Research In Motion	Blackberry 5820	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	160x160/1 bit
Samsung	SCH-X130	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	128x128/2 bits
Samsung	SCH-X230	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	120x160/8 bits
Samsung	SCH-X250	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	120x160/8 bits
Samsung	SCH-X350	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	128x128/2 bits
Samsung	SGH-S100	GSM/GPRS	900, 1800,	MIDP 1.0, CLDC 1.0	128x160/16 bits

			1900		
Samsung	SPH-A500	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0	128x128/12 bits
Samsung	SPH-N400	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0	128x96/16 bits
Samsung	SPH-X4209	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	128x160
Sanyo	A3011SA	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	132x176/16 bits
Sanyo	SCP-4900	AMPS, CDMA2000 1X	800, 1900	MIDP 1.0, CLDC 1.0	120x96/12 bits
Sharp	J-SH07	PDC	1500	MIDP 1.0, CLDC 1.0	120x160/16 bits
Sharp	J-SH08	PDC	1500	MIDP 1.0, CLDC 1.0	122x162
Sharp	J-SH51	PDC	1500	MIDP 1.0, CLDC 1.0	122x162
Siemens	C55	GSM/GPRS	900, 1800		
Siemens	M46	GSM/GPRS, TDMA	800, 900, 1900		
Siemens	M50	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	101x64/1 bit
Siemens	SL42	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	101x80/1 bit
Sony Ericsson	A3014S	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	120x120/16 bits
Sony Ericsson	P800	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0, PersonalJava	208x320/12 bits
Sony Ericsson	SO503i	PDC	800	CLDC 1.0	128x128/16 bits
Sony Ericsson	SO503iS	PDC	800	CLDC 1.0	128x128/16 bits
Sony Ericsson	SO504i	PDC	800	CLDC 1.0	128x128/16 bits
Toshiba	A3013T	CDMA2000 1X	800	MIDP 1.0, CLDC 1.0	144x176/16 bits
Toshiba	C5001T	CDMA	800	MIDP 1.0, CLDC	144x176/12
Toshiba	J-T06	PDC	1500	MIDP 1.0, CLDC	16 bits

Source: [Sun Microsystems]

Appendix D – The Accompanying CD-ROM

The accompanying CD-ROM contains the electronic version of this thesis in both Microsoft Word 2000 format "Thesis.doc" and Adobe Acrobat 4 (PDF1.3) format "Thesis.pdf", source codes, video clip demos, and other related material.