Widgets for Mobile Devices

GRADUATE PROJECT TECHNICAL REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by
Prashanth Gurrala

Fall 2008

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. David Thomas
Committee Member

Dr. Long-zhuang Li
Committee Member
Widgets for Mobile Devices

GRADUATE PROJECT TECHNICAL REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by
Prashanth Gurrala
Fall 2008

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. David Thomas
Committee Member

Dr. Long-zhuang Li
Committee Member
ABSTRACT

A Widget is a fancy looking tiny application which makes it simpler to perform some action. It does a simple thing but does it well. Mobile widgets can maximize screen space use and may be especially useful in placing live data-rich applications on the device home-screen/"phone-top”.

Using Microsoft Visual Studio for mobile Applications and .NET Compact framework, developers can associate the widgets and the Web services, where the widgets consume the Web services to retrieve information that can be displayed on the screen. These Web services act as an interface between the user and the corporate database (Information like the current weather, stock exchange, RSS feeds etc.).

Widgets are very good looking and can make work easier as they reside on the screen always. Instead of ending up with wallpapers, mobile devices look prettier with Widgets.
# TABLE OF CONTENTS

Abstract .......................................................................................................................... ii

Table of Contents ........................................................................................................ iii

List of Figures ............................................................................................................... vi

1. Background and Rationale .................................................................................. 1

1.1 Widget History ................................................................................................. 1

1.2 Types of Widgets ............................................................................................. 2

1.2.1 Desktop Widgets ......................................................................................... 2

1.2.2 GUI Widgets .............................................................................................. 3

1.2.3 Web Widgets .............................................................................................. 4

1.2.4 Mobile Widgets ......................................................................................... 4

1.3 Technical Background ..................................................................................... 5

1.4 Importance of Web Technology ...................................................................... 6

1.4.1 Role of Ajax ............................................................................................... 7

2. Narrative ............................................................................................................. 8

2.1 Widget 1.0 Requirements ............................................................................... 8

2.1.1 Widget API ............................................................................................... 9

2.2 What Makes Mobile Widgets an Ubiquitous Mobile Web? ......................... 9

2.3 Relation Between Mobile Ajax and Mobile Widget .................................... 10

2.4 Widgets and Gadgets ..................................................................................... 11

3. System Design ................................................................................................... 13

3.1 Software Requirements ................................................................................. 13
3.1.1 Visual Studio 2005……………………………………………...13
3.1.2 Microsoft Compact Framework (MCF)…………………………14
3.1.3 Windows Mobile Device Center (WMDC)…………………….14
3.1.4 Windows Mobile 5.0 Software Development Kit (SDK)………..14
3.1.5 Device Emulator Manager (DEM)……………………………...15

3.2 Widgets Design Architecture…………………………………………….15

3.3 General Development Procedure Steps…………………………….16

3.4 Widgets Development Procedure……………………………………..18

3.4.1 Weather Bug……………………..…………………….………...18
  3.4.1.1 Weather Bug Development Procedure………………………18

3.4.2 Currency Converter………………………………………………19
  3.4.2.1 Currency Converter Development Procedure…………….19

3.4.3 RSS Agent………………………..…………………….………..20
  3.4.3.1 RSS Agent Development Procedure……………………20

3.4.4 Gas Price Finder Widget………………………………………....20
  3.4.4.1 Gas Price Finder Development Procedure…………...……20

3.4.5 Games Information Widget……………………………………….21
  3.4.5.1 Games Information Development Procedure………………21

3.4.6 Stock Quote Widget……………………………………….……….21
  3.4.6.1 Stock Quote Development Procedure…………………….21

3.5 Environment Setup for Deployment……………………………………22

3.6 Deploying Applications to Emulator……………………………...23
  3.6.1 Mapping Widgets with Emulator……………………………..23

3.7 Consuming Web Service…………………………………………….24
3.7.1 SOAP

4. Testing and Evaluation

4.1 Tests with Emulators

4.2 Accuracy and Ease of Working with Emulators

4.3 Testing Weather Bug Widget

4.4 Testing Currency Converter Widget

4.5 Testing RSS Feed Widget

4.6 Widget Output Screens

5. Conclusion and Future Work

Bibliography and References

Appendix A. Terminology

Appendix B. Application Code
LIST OF FIGURES

Figure 1.1. Desktop Widget.................................................................3
Figure 1.2. GUI Widget.................................................................3
Figure 1.3. Web Widget.................................................................4
Figure 1.4. Mobile Desktop.............................................................5
Figure 1.5. Mobile Screen..............................................................5
Figure 2.1. Widget Technology-Stacks...............................................8
Figure 2.2. Widget.................................................................12
Figure 2.3. Gadget.................................................................12
Figure 3.1. Widget Request/Response Architecture............................16
Figure 3.2. Selecting New Project Type..........................................17
Figure 3.3 Add Web Service to Widget..........................................17
Figure 3.1. Basic Mobile Web Architecture........................................16
Figure 3.2. Page Handling Events.................................................17
Figure 3.3. Add Web Service to Widget..........................................17
Figure 3.4. Sample Code to Call Web Service................................18
Figure 3.5. Logic Implementing Weather Widget...............................19
Figure 3.6. Starting Emulator from Device Emulator Manager..............23
Figure 3.7. Cradle the Emulator to Connect to Internet.......................23
Figure 3.8. Mapping Widgets Folder in Emulator Properties................24
Figure 3.9. Emulator Image After Deploying Widgets.........................25
Figure 4.1. Mobile Device Center in Connected Mode........................27
Figure 4.2. Enter the Zip Code in Widget.................................................................29
Figure 4.3. Weather Widget Displaying Weather Information.................................30
Figure 4.4. Displaying Converted Amount on Currency Converter Widget..............31
Figure 4.5. RSS Feed With URL to Browse............................................................32
Figure 4.6. RSS Widget Loading the Information..................................................33
Figure 4.7. RSS Widget Loaded With Available Articles ......................................33
Figure 4.8. RSS Widget After Loading the News Article in New Window..............34
Figure 4.9. Screen Shot of Stock Quote Widget....................................................35
Figure 4.10. Screen Shot of Game Information Widget ........................................36
Figure 4.11 Screen Shot of Gas Price Information Widget ..................................37
1. BACKGROUND AND RATIONALE

The purpose of this project is to combine information available on the Web with platform services that include user experience and enhance user interface. Widgets integrate security while accessing platform services and provide additional features using cutting edge technologies like Windows Compact Framework (WCF) and Web services. Web services help to connect with corporate data using Web methods. Widgets use these Web methods to communicate with outside world via the Internet and display information on a mobile device.

1.1 Widget History

Widgets are cool, slick, and playful. Many media companies are turning to widgets. The rise of widgets was caused by several factors including the adoption of Real Site Syndication (RSS), the expansion of the blogosphere, the growth of social networks, the fashion of self-expression and the democratization of the Web at large. Originally, the goal of widgets was to simply deliver a miniaturized version of a specific piece of content outside of the primary Web site. A classic early Widget is the flicker badge, which allowed users to show a preview of their photos. Clicking on the badge would lead to the Flicker user's profile page with all the user's photos [Kennedy 2007].

The Widget technology that is being used has 25 years of evolution history. The concept of widgets gave a hope to the invention of the graphical user interface for home use. It was Bud Tribble and Andy Hertzfeld who came up with a concept called desktop ornaments in 1981 for the original Macintosh operating system. These ornaments were later named accessories, and they included small computing functions such as a
calculator, notepad, or simple games within a single application. Later in 1996, a personal start page application was built in the Netscape browser. It was named Netscape Power Start and this included many of the dynamic Web page concepts like the latest e-mail, stock quotes, weather reports, and other pieces of data from the Web and desktop into a single page [Kennedy 2007].

Later in October 2000, Stardock released a new GUI engine for Windows named Desktop-X. It displayed the system information such as CPU and memory utilization, news tickers, and live updates from the Web. Later in 2003, Konfabulator was released by Arlo Rose and Perry Clarke as a shareware side project. Konfabulator had blended Web technologies such as JavaScript with the desktop rendering strengths of OS X.

1.2 Types of Widgets

There are many kinds of widgets available in the recent days which promise to provide the customers with a dashboard like experience. Based on the environment they reside on, Widget products can be classified into: Desktop Widgets, GUI Widgets, Web Widgets, and mobile Widgets [Lenehan 2007].

1.2.1 Desktop Widgets

Desktop widgets are those which reside on the desktops. They typically provide easy access to frequently used functions and sometimes provide visual information. To host widgets a desktop requires that a Widget engine be installed, which is a host software system for physically installing and executing on the desktop. The ease of development made widgets more attractive. The most heard about desktop widgets include yahoo widgets and dashboard widgets for Apple Macintosh computer users.
Widgets, in this case, are downloadable interactive virtual tools that provide services such as showing the user the latest news, the current weather, a dictionary, a map program, sticky notes, or even a language translator, among other things [Wikipedia 2007].

![Figure 1.1 Desktop Widget](image1)

1.2.2 GUI Widgets

Generally, in computer programming, Widgets are considered to be the interface between the computer user and the windows environment such as a window or textbox. The widgets are termed virtual to differentiate them with their physical counterparts, e.g. virtual buttons that can be pressed with the mouse cursor and physical buttons that are pressed with fingers. Programmers use widgets to build the graphical user interface like in figure 1.2 which has features like slide bar, check box, drop down box and progress bar which create interaction with application using these virtual levers to control the program flow [Wikipedia 2007].

![Figure 1.2 GUI Widget](image2)
1.2.3 Web Widgets

Web Widget is a platform independent chunk of code that can be installed and executed with the help of a Web browser. This application does not require any additional compilation as it implements the idea of re-usable code. Other applications like gadgets, module, Capsule, and badge etc., also implement the concept of an application block that implements the code reusability strategy. Web widgets can also be developed using Dynamic HTML, Adobe Flash and Java Script. Figure 1.3 refers to a blog that implements application blocks to create and maintain using toolbox on the browser [Wikipedia 2007].

![Figure 1.3 Web Widget](Web wag 2006)

1.2.4 Mobile Widgets

Mobile widgets are the clones of desktop widgets, where the working environment is a mobile device or a cell phone. These are data-rich applications which maximize the screen space on the device home page. Cell phones built using windows mobile platform can implement Ajax-based functionality. Ajax-based widgets can consume the Web services and display information rather than using the client server request-response interaction. All major cell phone companies are developing an
environment which supports this new Ajax-based technology. Figure 1.5 refers to the mobile screen with icons implementing various applications [Wikipedia 2007].

Figure 1.4 Mobile Desktop [Web wag 2006]

Figure 1.5 Mobile Screen [Blip Networks 2007]

1.3 Technical Background

Widgets- these little applications are created using the existing Web technologies like XHTML, CSS etc. The Widget is not loaded by typing in a URL in the browser, but it is a Web application that is directly installed in the phone. Nokia calls their Widget environment as the Web Run-Time. The important aspect of this run-time is that the run-time environment can do things a regular Web browser was not able to do before: widgets can get access to the device API and consequently can interface to the phone hardware such as GPS or camera and other applications like the calendar or contact manager [Rabin 2006].
1.4 Importance of Web Technology

Technologies that drive the Web community are Extended Hyper Text Mark-up Language (XHTML)/ Cascading Style Sheets (CSS)/JavaScript and Ajax. Those technologies are powering the most successful applications and services on the Web. Mobile versions are hard to implement using these technologies as many versions are emerging day by day. Web development should be modified into a stage where every Web application available on the net should also be implemented on the mobile phone. All the applications that are implanted on the desktop should also be implemented on the mobile devices. This kind of application development is possible only by using cutting edge technologies like Ajax, and .NET micro application development environment.

With Widgets, interaction with hardware is involved in major parts, thus improving the performance of both mobile device and application involved with that device. Nokia invented a new camera which captures the location using GPS and sends the information onto the picture frame and displays on the picture which proves the cutting edge technology involved with the photography. This instance can be related to Widget technology where the application can include the location of the mobile device using GPS technology and send the picture to a corresponding account on the Web for storage, while doing something else like reading the latest RSS feed in the news Widget [Lineman 2007].

All these cutting edge technologies like XHTML, CSS, and Ajax are supported by all universally available Widget engines and mobile run time environments. One more important aspect regarding these technologies is that they follow that principle of “develop once and deploy anywhere and any number of times” as they make use of
developing platforms that are supported by universally available mobile technologies and platforms. One more area of development includes developing wrappers that enable widgets to work on any kind of environment making just the variable adjustments and environment configurations where ever deployment is possible.

1.4.1 Role of Ajax

Ajax will act as the heart of the technology that implements the widgets. It is possible only with Ajax that implements sneaky post back where only the required object is sent to the server instead of loading the whole page to update the current need. In the creation of Widgets Ajax toolbox and other open standard technologies play a major role [Lenehan 2007].

Widgets will provide phone users with favorite applications that require very little data from the Web and transmit live data using Ajax. These tiny applications connect with Web services and other applications on phone and with other users. If developing widgets for mobile phones will be as easy as anticipated above, these widgets will be able to do what regular browsers on mobile devices have been trying to achieve in quite some years until now: make people actually use the mobile Web. Since the Ajax requests can happen asynchronously and in the background the user of the application doesn’t even need to know that the “Web” is being accessed right now. The mobile Web will become ubiquitous and invisible at the same time - ok, concededly this is a very philosophical view - but maybe not even too far away from the future reality [Lenehan 2007].
2. NARRATIVE

2.1 Widget 1.0 Requirements

As stated earlier, widgets are single purpose applications that can be updated and/or displayed on local Web and then packaged as a single application and downloaded and installed on a user’s machine or a mobile device. The making of widgets involves various file formats and specifications to declare a user interface, script, package, digitally sign and deploy their applications. They run on user agents called Widget engines, which mimic the behavior of Web browsers [Smith 2007].

Widget engines support HTML, IRI’s, UNICODE, ECMA Script, various DOM levels, and multimedia resources like the sounds and images. They provide Application Programming Interfaces (APIs), which are similar to the Web Browsers, for the development of widgets. The rise in Ajax–style development leads the Widget engines to support the XMLHttpRequest object for making asynchronous data requests over HTTP. Figure 2.1 illustrates a typical technology stack that Widgets make use of at various points of their lifecycle [Caceres 2007].

![Figure 2.1 Widget Technology-Stacks (Caceres 2007)](image)

Figure 2.1 Widget Technology-Stacks [Caceres 2007]
After the widgets are developed, all the resources to make a Widget run as a single application are bundled as a single package with a special extension. This package includes manifest resources that a Widget engine uses to automatically configure a Widget upon instantiation. When the package is served over HTTP, the Web browser associates the package with the suitable Widget engine based on the extension or the resources it is served with [Caceres 2007].

2.1.1 Widget APIs

Widget APIs are a set of programming interfaces that provide functionalities that are specific to the instantiated widgets. The ability for Widgets to perform actions beyond the security scope of Web Widgets is afforded by Widget-specific APIs.

2.2 What makes Mobile Widgets a Ubiquitous Mobile Web?

The Ajax on mobile devices makes the mobile Web applications ubiquitous. This concept of Ajax on mobile devices was supported by Nokia, when it announced the release of their S60 platform which would support widgets using the Webkit based browser under the hood. Many other browser vendors have done this before. The first company to implement a working mobile Widget engine was Opera back in October 2004, it was called Opera Platform. While the Opera Platform was hard to install and more or less a proof-of-concept approach, Opera 9 for mobile, the browser that is soon to be released, also features widgets and comes pre-installed in many devices.

New versions of the Net front browser and the Open Wave Midas Project are announced to feature support not only for Ajax, but also for widgets. This shows that all major browser vendors in the mobile area have (finally) realized the importance of Ajax
and Widget support. The Apple’s Iphone features Ajax and the main way to add applications to the device is the widgets [Rabin 2006].

### 2.3 Relation between Mobile Ajax and Mobile Widget

Ajax comprises of XMLHttpRequest + DOM Scripting + XHTML/CSS. Functionally, two things are happening here: Data is being fetched asynchronously and the information thus fetched is being rendered on the client through JavaScript. Widget engines make use of various asynchronous rendering display parts to change the look of Widget rendering. Asynchronous I/O visual rendering is paving a new path to technologies where widgets are calling other widgets to perform certain functions. This leads to the fragmentation of Widget technology where Ajax renders the core technologies of underpinning widgets. This brings them to life taking control of each other [Lenahan 2007].

Two significant factors that led to the evolution of mobile widgets are:

1. Web Browsers supporting the mobile devices with Web technologies like JavaScript, CSS, Extended HTML, etc. Long tail applications are possible with Web mechanism and thus enabling the distribution of widgets. Soon this leads to the evolution of Widget framework where widgets call other widgets. Opera is already developing such platform to support Web technologies. Apple is also in the race of improving the dashboard of its Iphone.

2. To save the expensive mobile sync with Web services, Offline widgets are taking shape where the Widget goes offline when the connection is offline and later transfer data when connection is available [Smith 2007].
Mobile Widgets can be developed using two types of platforms depending on requirement of user:

1. Using Windows Compact Framework
2. Using Palm operating system

Mobile Widgets developed in current market can be classified as:

1. Widsets
2. S60 Widgets

Widsets are Java based applications developed using widest studio. Widset studio is simple visual studio editor which enables user to develop Widgets in minutes depending on its functionality. However they have their own limitations as they need RSS feeds to work on. To develop custom controls using the Widset studio Helium script is developed that implements java like coding [Lenehan 2007].

S60 Widgets are light weight applications developed using advanced Ajax and needs a mobile run-time to execute them.

2.4 Widgets and Gadgets

“A Widget is a small GUI application (both visually and computationally) that provides information on, or an interface to, a very restricted set of functionality or data.”

Widgets can be compared to automobile gages that display information regarding a particular function like tachometer, speedometer, gas level, etc [Monson 2007].

Definition for software widgets and gadgets:

“A small footprint application that provides relevant on-demand, personalized, encapsulated information from a predetermined data source.”
First Widget was released by Apple in 1984 which looked similar to the Figure 2.4

In 1996 PointCast introduced pre-cursor widgets with information from a specific data set. This technology is called “push technology”. This technology was replaced with an interface that provide data to applications like remote procedures [Monson 2007].

Figure 2.2 Widget [Monson 2007]

Figure 2.3 Gadgets [Monson 2007]
3. SYSTEM DESIGN

Building mobile internet applications needs Windows server running IIS (Internet Information Services) and .NET compact framework. The Microsoft mobile Internet Toolkit extends ASP.NET to easily target mobile devices using mobile Web forms technology, ADO.NET for data access. Production environment includes a Web server and mobile Web application that are deployed on the internet.

3.1 Software Requirements

Software requirements for this project include:

- Visual Studio 2005
- .NET compact framework provided by Microsoft
- Windows Mobile Device Center
- Windows Mobile 6.0 SDK (software development kit)
- Device Emulator Manager

Visual Studio 2005 comes with compact framework for supporting development of mobile applications. Mobile Device Center helps to synchronize mobile emulator with internet. Current project builds three Widgets - Weather bug, currency converter and RSS feed which navigates pages available on any Web site that offer free consumption of RSS feed.

3.1.1 Visual Studio 2005

VS 2005 is the Integrated Development Environment which includes tools to debug, compile and run applications. It also includes inbuilt toolkit which makes the
development easy. It also integrates compact framework to support mobile application
development.

3.1.2 Microsoft Compact Framework (MCF)

Compact Framework is a smart device development framework which provides
managed code and Web services to a developer. This framework supports many
platform-independent protocols like XML, SOAP and HTTP which targets mainly on
resource-constrained devices like smart phones and PDA’s. MCF simplifies the process
of creating and deploying mobile applications while taking full advantage of device
capabilities. MCF is embedded into Visual Studio 2005 development environment
[Microsoft 2007].

3.1.3 Windows Mobile Device Center (WMDC)

WMDC provides device management features for Windows mobile powered
devices. Some of the features include:

- Streamlined setup for improved management
- Robust synchronization of calendars, contacts, favorites and files
- Media synchronization for multimedia files
- File browsing enables to quickly locate documents on device

3.1.4 Windows Mobile 5.0 Software Development Kit (SDK)

Visual Studio 2005 extends Mobile SDK so that the developer can write managed
code targeting mobile 5.0 Pocket PC devices. Some of the features include:

- Device emulator images and skin files.
- Tools and configuration files.
### 3.1.5 Device Emulator Manager (DEM)

DEM is tool which helps to launch emulator for testing mobile applications on a computer. DEM is integrated into Visual Studio 2005 after installing mobile SDK. This tool is auto launched when a mobile application debugged after building solution. DEM includes all the emulator skins provided by SDK, to select and deploy the application for testing. Once the emulator is up and running, the internet can be accessed by using the option called ‘cradle’ which enables to share desktop Internet connection.

### 3.2 Widgets Design Architecture

Request – Reply behavior of emulator consuming a Web service is shown in Figure 3.1. Mobile emulator acts as an external device connected to the computer using a cable. Widget references to the service, which it has to call, and sends the User input along with its request. This request is carried to the client via Internet. Client processes this information and returns data using HTTP protocol. This data is displayed on the device in appropriate fields.

Device Emulator Manager acts as an interface to connect emulator with host computer. DEM enables emulator to connect with Internet on computer by referencing mobile device center. Device center also enables to change connection preferences of emulator, file management and control management.
3.3 General Development Procedure Steps

Widget development using .NET technologies involves standard procedure besides implementing logic. General development procedure includes following steps

1) In Visual Studio 2005, create new project of type Windows mobile 6 device application and name the project.
2) Open the Form to design User Interface by including required controls on the screen.

3) Add Web reference of service which includes the methods to access data from corporate database.
4) Using Web reference name, instantiate the methods and pass the user input values.

```
WebReferenceNAME.serviceMETHOD serviceOBJECT = new
WebReferenceNAME. serviceMETHOD.getServiceMETHOD();
returnOBJECT = WebReferenceNAME. getServiceMETHOD
(userInputPARAMETER);
    if (returnOBJECT!= null)
    {
        Display contents on Widget;
    }
```

Figure 3.4 Sample Code to Call Web Service

### 3.4 Widgets Development Procedure

Using the software requirements mentioned in 3.1, the following Widgets are developed.

#### 3.4.1 Weather Bug

This Widget consumes a Web service from http://www.weather.com and sends the zip code information as the input parameter to retrieve the weather information like city, state, time and temperature.

#### 3.4.1.1 Weather Bug Development Procedure

Weather Bug application development includes the following procedure:

1) In VS 2005 create a new project.

2) Add the label and text box controls to form designer.

3) In the item click event, capture the zip code value from text box and append it to query string of SOAP message.

4) Call the Web service of [www.weather.com](http://www.weather.com)
5) Display appropriate xml tags returned by Web service in the text box.

```csharp
string serviceUrl = "http://xoap.weather.com/weather/local/" + textBox1.Text + "?cc=*&dayf=5&link=xoap&prod=xoap&par=1072974744&key=03983e4b39790278";
HttpWebRequest _webreq = null;
HttpWebResponse _webresponse = null;
Stream _stream = null;
StreamReader _streamreader = null;
try
{
    _webreq = (HttpWebRequest)WebRequest.Create(serviceUrl);
    _webreq = (HttpWebResponse)_webresponse.GetResponse();
    _stream = _webresponse.GetResponseStream();
    _streamreader = new StreamReader(_stream);
    serviceUrl = "";
    serviceUrl = _webresponse.ReadToEnd();
    XmlDocument xd = new XmlDocument();
    xd.LoadXml(serviceUrl);
    string[] temp = xd.SelectSingleNode("/weather/loc/dnam").InnerText.ToString().Split('\\');
    textBox2.Text = temp[0];
    textBox3.Text = xd.SelectSingleNode("/weather/cc/tmp").InnerText.ToString();
    textBox5.Text = xd.SelectSingleNode("/weather/cc/t").InnerText.ToString();
}
```

Figure 3.5 Logic implementing Weather Widget

### 3.4.2 Currency Converter

This Widget consumes a Web service and sends the amount value and currency types as the input parameter to retrieve the current exchange value. This Widget can also be used for foreign travel expense calculation.

#### 3.4.2.1 Currency Converter Development Procedure

Currency Converter application development includes the following procedure:

1) In VS 2005 create a new project.
2) Add the label, drop down box and text box controls to form designer.

3) In the item click event, capture the currency types and amount value from text box and append it to query string of SOAP message.

4) Call the Web service of www.xignite.com

5) Display appropriate xml tags returned by Web service in the text box.

3.4.3 RSS Agent

This Widget consumes a RSS feed which contains the information in XML format. It then gives an option to select an article and browse to that article.

3.4.3.1 RSS Agent Development Procedure

RSS Agent application development includes the following procedure:

1) In VS 2005 create a new project.

2) Add the drop down box and text box controls to form designer.

3) In the item click event, capture the RSS link from drop down menu.

4) Append the link to the browser to get the news information and redirect to that page.

3.4.4 Gas Price Finder Widget

This Widget consumes a Web service from local machine and sends the zip code information as the input parameter to retrieve the Price information along with city.

3.4.4.1 Gas Price Finder Development Procedure

Gas Price application development includes the following procedure:

1) In VS 2005 create a new project.

2) Add the label and text box controls to form designer.

3) In the item click event, capture the zip code value from text box and append it to service method.
4) Call the Web service

5) Display appropriate xml output values on the Widget.

3.4.5 Games Information Widget

This Widget consumes a Web service from local machine and sends the game type information as the input parameter to retrieve the game details like date, teams playing the game and telecasting channel.

3.4.5.1 Games Information Development Procedure

Games information Widget development includes the following procedure:

1) In VS 2005 create a new project.

2) Add the label and text box controls to form designer.

3) In the item click event, capture the game type value from drop down list and append it to service method.

4) Call the Web service

5) Display appropriate responses on the Widget.

3.4.6 Stock Quote Widget

This Widget consumes a Web service from local machine and sends the company code as the input parameter to retrieve the company details like name, previous quote value and current quote value.

3.4.6.1 Stock Quote Development Procedure

Stock Quote Widget development includes the following procedure:

1) In VS 2005 create a new project.

2) Add the label and text box controls to form designer.
3) In the item click event, capture the company code value from text box and append it to service method.

4) Call the Web service

5) Display appropriate responses on the Widget.

3.5 Environment Setup for Deployment

After building the applications and converting it into executables, set up environment requires the following steps:

- Start the Windows Mobile Device Center (WMDC)
- Start the Device Emulator Manager (DEM)
- Select the emulator image from DEM and click the connect option from its menu
- After loading the emulator with default settings right click on its image in DEM and select the Cradle to establish connection with Internet from computer
Figure 3.6 Starting Emulator from Device Emulator Manager

Figure 3.7 Cradle the Emulator to Connect to Internet
3.6 Deploying Applications to Emulator

Once the emulator is connected to Internet, it is ready to deploy applications and test. Deploying applications is just like copying applications into mobile device using a data cable and synchronizing software.

3.6.1 Mapping Widgets with Emulator

To deploy Widgets onto emulator configure the emulator settings, to map the file containing Widgets. Browse to the ‘General’ tab in the properties window to select the address of folder containing Widgets. This procedure is shown in figure 3.8 Mapping Widgets Folder in Emulator Properties.
3.7 Consuming Web Service

Web services are one of the interesting technologies to emerge recently. It’s an entity that can be programmed to provide a particular functionality to any number of applications over internet. They are similar to class libraries, but physically they reside on the remote server rather than on the application code. Client applications communicate with Web Services, sending them parameters to receive generated results in true black-box fashion. The protocol used in this kind of communication is Simple Object Access Protocol (SOAP), an XML-based mark-up language. The best feature of the Web services is that they do not carry any state information with them and are not affected by firewalls [Microsoft 2007].

It is good to use these Web services when,

- Using caching features of ASP.Net.
- Using SSL security or windows authentication.
- Integrating with third party system.

3.7.1 SOAP

Simple Object Access Protocol is an XML bases Mark-up language. It is used for messaging. A SOAP message is simple XML document, which always contains a mandatory envelope and body element and can contain few optional elements, such as header and fault elements. A SOAP Envelope is a set of rules for messages.
Figure 3.9 Emulator Image after Deploying Widgets
4. TESTING AND EVALUATION

4.1 Tests with Emulators

Emulators help to test wireless devices without buying them. Software emulation is the technique intended to support developers to test applications before implementing on target devices. Software Emulators can simulate the operation of a wireless device on existing mobile phone or desktop [Ruley 2006].

Emulators mimic the behavior of Windows based hardware platforms where the screen resolution can be set and also include a skin for the emulator. Skin changes the appearance and functionality of GUI for the emulator.

4.2 Accuracy and Ease of Working with Emulators

The manufacturers of all mobile devices provide emulators that simulate the operation of their hardware and browsers. Emulator software enables user to view mobile Web pages as they might appear on the manufacturers' hardware devices, and to experience the interface for Web site as users do. Developing and testing with emulators enables user to test your mobile Web application more easily on a variety of devices before deploying it.

4.3 Testing Weather Bug Widget

Testing this Widget is very simple and straight forward. The following steps are included in the testing process:

1) Start the Windows Mobile Device Center
2) Start the Device Emulator Manager and connect emulator

3) Cradle the emulator to connect with computer’s Internet service

4) Map the Widget location in emulator’s properties

5) Run the Widget by selecting it and enter the five digit zip code

6) Click on the ‘Show Weather’ button

7) Weather information is displayed on the text box

8) Click on ‘Exit Application’ to close the Widget

Mobile Device Center acts as the interface which connects emulator with outside world using the existing Internet connectivity on computer. This also helps to manage files on mobile device by browsing File Management section to add and delete files on device.

Figure 4.1 Mobile Device Center in Connected Mode
Activating Weather widget will bring the form in Figure 4.2. Enter the zip code into text box and click on “Get Weather” button. This will internally invoke the button click event of the form and execute the corresponding logic.

![Weather Widget](image)

Figure 4.2 Enter the Zip Code in Widget

Button click event will invoke the Web service by appending zip code entered by user. This request is transported via SOAP protocol using Internet to the server hosting Web service. Server in turn searches the real time weather information from its database and sends back response through the same SOAP protocol. This result is handled by .NET compact framework, converted into text format and displayed on the screen at appropriate text boxes.

This result form shown in Figure 4.3 can be used repeatedly by entering zip code on the result form and clicking “Get Weather” button to get weather information of different location. Press the “EXIT” button to close this form and return back to home screen.
4.4 Testing Currency Converter Widget

Testing this Widget is very simple and straightforward. The following steps include in testing process:

1) Start the Windows Mobile Device Center
2) Start the Device Emulator Manager and connect emulator
3) Cradle the emulator to connect with computer’s Internet service
4) Map the Widget location in emulator’s properties
5) Run the Widget and enter the amount of conversion
6) From the Drop down menu select the currency type
7) Converted amount is displayed in the text box
8) Click on ‘Exit’ to close the Widget
4.5 Testing RSS Feed Widget

Testing this Widget is very simple and straight forward. The following steps include in testing process:

1) Start the Windows Mobile Device Center  
2) Start the Device Emulator Manager and connect emulator  
3) Cradle the emulator to connect with computer’s Internet service  
4) Map the Widget location in emulator’s properties  
5) Run the Widget and enter the amount URL of RSS and click the ‘Browse RSS’ button  
6) From the Drop down menu select the news item to browse and click on ‘Read Article’ button
7) Article is either displayed on the text box or opened in a new browser window depending on the selected article properties

8) Click on ‘Exit’ to close the Widget

Figure 4.5 RSS Feed with URL to Browse

Select the available RSS feed address from drop down box and click on “Browse RSS” button to invoke the RSS. This will call the button click event handler to execute logic behind it. Make sure the mobile Device Center is in connected mode to access the Internet. This is indicated by the arrows pointing in opposite direction on the mobile screen header. Button click event will invoke the form browser to connect with RSS host server. This will in turn respond with XML tags associated with URL link. These links are populated into drop down box, so that user can select and read that article.
After selecting an article, click on “Read Article” button as shown in Figure 4.7. This will invoke the form browser sending the URL associated with article, to the hosting server.
Response from hosting server is displayed on new browser window of mobile emulator. Make sure this Widget also has an active Internet connection to browse through RSS feed.

![RRSS Widget after Loading the News Article in New Window](image)

**Figure 4.8 RSS Widget after Loading the News Article in New Window**

### 4.6 Screenshots of Widget Outputs

Activating Stock Quote widget will bring the form in Figure 4.9. Enter the company code into text box and click on “Get Quote” button. This will internally invoke the button click event of the form and execute the corresponding logic. Button click event will invoke the Web service by appending company code to service method. This request is transported via SOAP protocol using Internet to the server hosting Web service. Server in turn searches the real time stock information from its database and sends back response
through the same SOAP protocol. This result is handled by .NET compact framework, converted into text format and displayed on the screen at appropriate text boxes.

![Figure 4.9 Screen Shot of Stock Quote Widget](image)

Figure 4.9 Screen Shot of Stock Quote Widget

This result form shown in Figure 4.9 can be used repeatedly by entering company code on the result form and clicking “Get Quote” button to get stock information of different company. Press the “EXIT” button to close this form and return back to home screen.

Activating Sports Information widget will bring the form in Figure 4.10. Select the game from drop down list. This will trigger the item changed event handler of drop down box and execute the corresponding logic. This logic calls the Web service by appending game code to service method.
This request is transported via SOAP protocol using Internet to the server hosting Web service. Server in turn searches the real time game information from its database and sends back response through the same SOAP protocol. This result is handled by .NET compact framework, converted into text format and displayed on the screen at appropriate text boxes.

Activating Gas Price Info widget will bring the form in Figure 4.11. Enter the zip code into text box and click on “Get Price” button. This will internally invoke the button click event of the form and execute the corresponding logic. Button click event will invoke the Web service by appending company code to service method.
This request is transported via SOAP protocol using Internet to the server hosting Web service. Server in turn searches the real time price information from its database and sends back response through the same SOAP protocol. This result is handled by .NET compact framework, converted into text format and displayed on the screen at appropriate text boxes.
5. CONCLUSION AND FUTURE WORK

Emerging mobile technologies helps companies to manufacture mobile phones which are more potentially equipped. The advancement in technologies helped user to access the World Wide Web through these devices. Mobile Widgets are the tools that have great potential to display the contents on WWW on the platform. These serve the user more efficiently to access the World Wide Web rather than the user interface provided by the phone architecture.

Widgets interact with the Web service content in an easier and faster way than the regular applications. They help the user to manage the day to day Web content like weather, stock exchange, flight travel information, directions, and translations. They help the user to achieve all these applications in a good looking portable locus and with minimal processing resources. Widgets developed on mobile devices can also be implemented on desktop computers, thus making the code reusable for the developer. It has become easier to develop an application and deploy it on a platform because of the availability of emulators from all the major mobile manufacturing companies.

Widgets can be easily downloaded into the mobile devices, as they do not require any further installation equipment. They are just a piece of software applications which allows user to browse through WWW on his finger tips.


APPENDIX A. TERMINOLOGY

- **Ajax**: Asynchronous Java script and XML
- **DEM**: Device Emulator Manager helps the Emulator to interact with computer.
- **Emulator**: Software version of a Mobile device which performs like a real device when connected with computer.
- **RSS**: Real Site Syndication. RSS is a XML format application to store information.
- **SDK**: Software Development Kit is a package which includes necessary development toolkit.
- **SOAP**: Simple Object Access Protocol is used for transferring data to/from mobile device to Web service.
- **WCF**: Windows Compact Framework. Microsoft framework which supports mobile device programming.
- **Web Service**: A service transmitting data which can be accessed though consuming it with service methods.
- **Widget**: Graphical application on mobile device which does an action.
- **WMDC**: Windows mobile Device Center helps to connect Emulator with Internet
- **XML**: Extended Mark-up Language
// Logic implementing Currency Converter Widget

CurrencyWidget.Header objHeader = new 
CurrencyConverterAebService.CurrencyWidget.Header();
objHeader.Username = "pgurrala1@islander.tamucc.edu";

CurrencyWidget.XigniteCurrencies objCurrencyWidget = new 
CurrencyConverterAebService.CurrencyWidget.XigniteCurrencies();
objCurrencyWidget.HeaderValue = objHeader;
CurrencyWidget.CrossRate objCrossRate;
objCrossRate =
objCurrencyWidget.GetRealTimeCrossRateAsString(comboBox1.SelectedItem.ToString(),comboBox2.SelectedItem.ToString());
if (objCrossRate == null)
{
    textBox1.Text="Service is unavailable at this time.";
}
else
{
    textBox1.Text = Convert.ToString((Convert.ToDouble(textBox1.Text)
* Convert.ToDouble(objCrossRate.Rate)));
}

// Logic implementing RSS Widget

switch (MenuItem2.Text)
{
    case "Browse RSS":
        textBox2.Visible = false;
        string ay = ComboBox1.Text;
        if (ay == "") { ay = textBox2.Text; }
        if (ay.Substring(0, 7).ToLower() == "http://")
        {
            Label1.Text = "Browsing RSS" + vbCrLf + ay;
            ComboBox1.Visible = false;
            Label1.Visible = true;
            MenuItem2.Enabled = false;
            TextBox1.Visible = true;
            //Download the rss info here...
            TextBox1.Text = "Loading URL " + ay + "... Please wait.";
            Application.DoEvents();
            HttpWebRequest wr;
            HttpWebResponse ws;
try {
    wr = (HttpWebRequest)WebRequest.Create(ay);
    ws = (HttpWebResponse)wr.GetResponse();
}
catch (Exception eek) {
    MessageBox.Show(eek.Message);
    break;
}
ComboBox1.Items.Clear();
ComboBox1.Items.Add(" Select Item to Read ");
System.Xml.XmlDocument xd = new System.Xml.XmlDocument();
xd.LoadXml(ay);
foreach (System.Xml.XmlNode xn in xd.SelectNodes("/rss/channel/item")) {
    System.Xml.XmlDocument xd1 = new System.Xml.XmlDocument();
    xd1.LoadXml(xn.OuterXml);
    RssItem rsi = new RssItem(xd1.SelectSingleNode("/item/title").InnerText,
        xd1.SelectSingleNode("/item/description").InnerText,
        xd1.SelectSingleNode("/item/link").InnerText);
    ComboBox1.Items.Add(rsi.Title);
    RSSSet.Add(rsi);
}
TextBox1.Text = "RSS data loaded. Select a title from the list above to view the content here. Some RSS items may need a browser to be read, and will be opened accordingly.";
} else {
    MessageBox.Show("URL Invalid");
    break;
}
case "Read Article":
    if (MessageBox.Show("Browse this site? " + vbCrLf + (((RssItem)RSSSet[ComboBox1.SelectedIndex - 1]).Description + " ").Substring(0, 250).Trim() + vbCrLf, ((RssItem)RSSSet[ComboBox1.SelectedIndex - 1]).Title,
MessageBoxButtons.YesNo, MessageBoxIcon.None, MessageBoxDefaultButton.Button1) == DialogResult.Yes)
{
    MessageBox.Show("RssReader will exit after starting this link in your browser");
    Process pr;
    ProcessStartInfo sifu = new
    ProcessStartInfo("\windows\iexplore.exe", " " + ((RssItem)RSSSet[ComboBox1.SelectedIndex - 1]).Link);
    sifu.UseShellExecute = false;
    sifu.WorkingDirectory = "\windows";
    pr = Process.Start(sifu);
}
break;
}

// Logic implementing Gas Price Finder Widget
refGasPrice.GasPriceWidget objGasPrice = new
GasPriceWidger.refGasPrice.GasPriceWidget();
refGasPrice.GasPriceInformation objGasPriceInfo = new
GasPriceWidger.refGasPrice.GasPriceInformation();
if (objGasPriceInfo != null)
{
    textBox2.Text = objGasPriceInfo.CityName.ToString();
}

// Logic implementing Game Information Widget
SportsInfo.SportsInformation objSports = new
SportsInfo.Information objGetInfo = new
SportsWidget.SportsInfo.Information();
objGetInfo = objSports.GetSportsInfo(cmbGames.SelectedItem.ToString());
if (objGetInfo != null)
// Logic implementing Stock Information Widget

getStockQuotes.Service1 objStockInfo = new Stock_Widgetsingle.getStockQuotes.Service1();
getStockQuotes.StockInformation objGetInfo = new Stock_Widgetsingle.getStockQuotes.StockInformation();
objGetInfo = objStockInfo.GetStockInfo(textBox1.Text.ToString().ToUpper());
if (objGetInfo != null)
{
    textBox2.Text = objGetInfo.CompanyName.ToString();
    textBox3.Text = objGetInfo.CurrentQuote.ToString();
    textBox4.Text = objGetInfo.PreviousQuote.ToString();
}