

國立交通大學

資訊科學與工程研究所

碩士論文

一個提供 Widget 應用程式的跨平台環境

A Cross-platform environment for mobile Widget-based
application

研究生：楊貴安

指導教授：袁賢銘 教授

中華民國 九十九 年 六月

一個提供 Widget 應用程式的跨平台環境
A Cross-platform environment for mobile Widget-based application

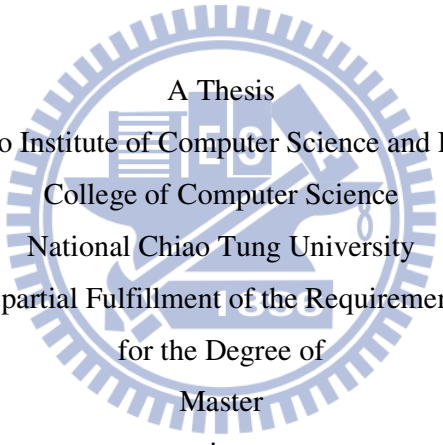
研究生：楊貴安

Student : Kuei-An Yang

指導教授：袁賢銘

Advisor : Shyan-Ming Yuan

國立交通大學
資訊科學與工程研究所
碩士論文



A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

一個提供 Widget 應用程式的跨平台環境

國立交通大學資訊科學與工程研究所

摘要

對於現階段的行動裝置而言，Widget 類型的應用程式所適用的平台環境，基本上尚分為兩種類型：第一種就是此平台的應用程式只限於自己的系統，並無法移植到其他的平台，例如：在 Windows Mobile 所開發的 Widget。另一種類型則可提供一個共用的平台供程式開發與執行，例如：Yahoo! Go。Yahoo! Go 之所以能夠支援許多的行動裝置，是因為它必須花費許多時間及技術去克服大量的相容性問題。因為這樣的設計概念必須伴隨著平台環境的異動而去做定期的更新以及修正，甚至是大幅度的改變。

Apache Shindig 是一個透過伺服器端執行的網路應用程式平台，透過它能將 Widget XML 轉成 HTML 和 JavaScript 的程式碼。另一方面，Google Gears 技術使得離線應用程式的運作變成可能。為了去製造一個跨平台且易於維護的行動應用程式環境，本論文擬將此兩種特性結合，以 Widget 為基礎開發一個行動應用程式的執行環境。

關鍵詞：行動應用程式平台、Apache Shindig、Google Gears

A Cross-platform environment for mobile Widget-based application

Student: Kuei-An Yang

Advisor: Shyan-Ming Yuan

Department of Computer Science and Engineering
National Chiao Tung University

Abstract

Today, mobile application platform for Widget can basically be divided into two types: One is to develop products for its own system only; application is not portable to other platforms, such as: Windows Mobile Widgets.... Another is to provide a common platform for applications development, such as: Yahoo! GO. Yahoo! Go is able to support many mobile platforms, because Yahoo must spend a lot of time and invest heavily in technical resources to overcome many compatibility problems. However, this design concept will make the platform must be accompanied the transaction of environment to do regular updates and corrections, even substantial changes.

Apache Shindig is a server-side-based Web application platform and can be rendered Widget XML into html code. On the other hand, the Google Gears technology makes the offline application execution possible. In order to create a Cross-Platform environment which can be easily maintained, in this thesis, we will integrate these two functionalities together, and propose a Web-based mobile application execution desktop environment.

Keywords : mobile application platform, Apache Shindig, Google Gears

Acknowledgement

歷經了兩年的修業,最後我也終於來到畢業這最後的關卡,腦袋裡依稀還記得兩年前,那有些青澀的自己,面臨遲到的最後一刻來到袁老師的實驗室,敲打著研究室的門..第一次來到實驗室,看到落地窗外優美的景色所帶給我的震撼...如今,求學生涯不順遂的我也終究種出了這苦澀的果實。

再多的話語也不能夠形容我對於老師的景仰,會議中自己很認真聽都不太明白的研究,老師卻還能從中挑出方向,給與我做學問所需要的思維。另外,實驗室中的眾多強者學長,也實在地讓我體驗到創業的重要,而且興趣與工作是可以結合的!

感謝這兩年來的實驗室伙伴 昱凱帶著我進入各個課堂中的前兩排座位,並且與我一起衝刺了許多漫漫長夜。感謝 國亨總是在我遇到 Bug 時,耐心地聽完我的敘述,並且給我適當的建議。感謝 銘鴻規劃了如此多的活動,讓我適當地放鬆自我,並且忍受我作息不大正常的生活習慣。感謝 大眼帶給我不同於自己的處事想法,也感謝 Fang 陪我練習了這麼多的英文會話,並且在我聽不懂時,很有耐心的換句話對我說。

我還要感謝我的女朋友 毓君,因為你那認為我絕對做的到的態度帶給我自信,在我筋疲力盡的時候,貼心的主動幫我按摩讓我有動力持續前進。最後,我要感謝我的爸媽,在我這段稍些漫長的求學生活中,給了我這麼大的支持,無論是經濟上的支援或是心理上的鼓勵,你們從不曾刻求的態度,才能讓我在退伍之後還有讀書的動力。雖然花的時間久了些,不過,還是對您們說聲:「爸、媽,我做到了!」

Table of Contents

Acknowledgement.....	ii
Table of Contents.....	iii
List of Figures	v
List of Tables.....	viii
Chapter 1 Introduction.....	1
1.1 Preface.....	1
1.2 Motivation	3
1.3 Research Objectives.....	4
1.4 Research Contribution	6
1.5 Outline of the Thesis.....	7
Chapter 2 Background	8
2.1 Google Gadget.....	8
2.1.1 Introduction of Google Gadget.....	8
2.1.2 Main APIs of Google Gadget	9
2.1.3 A sample of Google Gadget	9
2.2 Google Gears	10
2.2.1 Introduction of Google Gears	10
2.2.2 System architecture and off-line theory of Google Gears	11
2.2.3 The information security of Google Gears	16
2.2.4 The functions of Google Gears API.....	16
2.3 Apache Shindig.....	17
2.3.1 What is Apache Shindig?	17
2.3.2 Five different servlets in Apache Shindig	18
2.4 Related Works	19
2.4.1 Yahoo! Go	19
2.4.2 Web Page Tailoring Tool	21
2.4.3 HTML 5.....	23
Chapter 3 System Design.....	25
3.1 Overview	25
3.2 System Architecture.....	27
3.2.1 Mobilize Service	29

3.2.2 RSS Handler/Parsing Widget.....	31
3.2.3 Sync Service	32
3.3 The behavior of Mobile device.....	32
3.4 Summary	33
Chapter 4 System Implementation	35
4.1 Application Development	35
4.1.1 Sequence Diagram of Application Development.....	35
4.1.2 Implement details of Rewriting Widget	36
4.2 User Subscription.....	38
4.2.1 Sequence Diagram of User Subscription.....	38
4.3 Implementation of the two demonstrations.....	39
4.3.1 Sequence Diagram of To-Do List	39
4.3.2 Implementation of Sync Service.....	40
4.3.3 Sequence Diagram of RSS Widget	42
4.4 The other problems we encountered.....	45
4.5 Implementation of caching Remote WebPages.....	47
4.6 Summary	49
Chapter 5 System Demonstration & Evaluation.....	50
5.1 Basic Operation	50
5.1.1 Developer uploads widgets.....	50
5.1.2 User selects widgets	51
5.1.3 Do operation in mobile phone.....	53
5.2 The Demo of To-Do List.....	55
5.2.1 Functional test and Synchronization test in on-line state	55
5.2.2 Test the off-line operation	57
5.3 The Demo of RSS type Widget	61
5.4 System Evaluation	63
Chapter 6 Conclusion and Future Work.....	66
Reference and Bibliography	68
Appendix A:	71

List of Figures

Figure 1-1: To-Do List.....	5
Figure 1-2: RSS type of Widget.....	6
Figure 2-1: Sample of Google Gadget	9
Figure 2-2: Traditional requests	12
Figure 2-3: Join the data layer	12
Figure 2-4: Data Switch layer	13
Figure 2-5: Database of Google Gears	13
Figure 2-6: Sync Engine of Google Gears.....	16
Figure 2-7: The Class diagrams of these five servlets.....	19
Figure 2-8: Yahoo! Go software.....	20
Figure 2-9: Page Tailor in Web browser.....	22
Figure 3-1: Overview	26
Figure 3-2: System Architecture.....	29
Figure 3-3: Web Page Tailoring System.....	31
Figure 3-4: Combined with our system.....	32
Figure 3-5: Display of Tailor	33
Figure 3-6: RSS Handler	33
Figure 3-7: Sync Service	34
Figure 3-8: The behavior of mobile device	35
Figure 4-1: Sequence Diagram of Application Development.....	37
Figure 4-2: Cache list	39
Figure 4-3: Sequence Diagram of User Subscription.....	40
Figure 4-4: Sequence Diagram of on-line To-Do List.....	41
Figure 4-5: Sequence Diagram of off-line To-Do List.....	42

Figure 4-6: Prefs function.....	42
Figure 4-7: Implementation of Save/Load.....	43
Figure 4-8: Sequence Diagram of on-line RSS Widget	44
Figure 4-9: Sequence Diagram of off-line RSS Widget.....	45
Figure 4-10: Add off-line link.....	45
Figure 5-1: Developer uploads the widget file	52
Figure 5-2: Uploaded.....	53
Figure 5-3: The user's Homepage.....	54
Figure 5-4: User Management Interface.....	54
Figure 5-5: The user's selection completed.....	55
Figure 5-6: Users' Homepage in mobile device.....	56
Figure 5-7: Users' Widget Interface.....	56
Figure 5-8: User allows Google Gears	57
Figure 5-9: To-do-List Widget in desktop browser.....	58
Figure 5-10: Add new items on mobile browser.....	58
Figure 5-11: Completed the add action	58
Figure 5-12: Real time synchronization	59
Figure 5-13: Setting toolbar of mobile phone.....	60
Figure 5-14: The network status of mobile phone	60
Figure 5-15: The network of mobile phone has been shut down.....	60
Figure 5-16: Offline users' interface	61
Figure 5-17: Demo of To-Do-List widget	61
Figure 5-18: Functional testing.....	62
Figure 5-19: Connection Setting	62
Figure 5-20: After synchronization	63
Figure 5-21: On-line Request Message	64

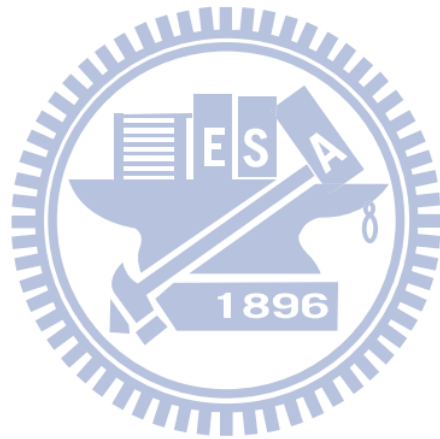
Figure 5-22: Cache finish Message..... 64

Figure 5-23: Go to the remote webpage 64

Figure 5-24: Shows the real content..... 64

Figure 5-25: Off-line Request Message 65

Figure 5-26: Shows the tailored content..... 65



List of Tables

Table 4-1: Comparison table of Mobilize effect	47
Table 5-1: Comparison table of different platform	66
Table 5-2: Means and standard deviations from the “Usability Score”	67



Chapter 1 Introduction

1.1 Preface

With the performance's promotion of mobile devices, and the progress of wireless communication technology, mobile network-based software in the handheld mobile device is not only feasible; it began to be taken seriously. We can imagine the future, more and more of the traditional web software gradually migrates to mobile devices to meet the business, life, entertainment, and many other needs.

However, in wireless networks based, besides the feasibility of developing Web applications [1], there are many issues that need to be considered: first, mobile device software must be installed on hundreds of different operations, different platforms, it makes deployment and distribution for the application software more difficult. Presently, the market position of the platform, the more there is Java ME [2], Windows Mobile [3], Symbian [4], BREW [5], Mobile Linux [6], and the Google open platform Android [7]. Such diversity of platforms and with the development model gave software developers more difficulties.

In addition, network-based applications for mobile devices are vulnerable to the surrounding environment, such as telecommunications services, it is easy to receive network connectivity problem caused by terrain, buildings, rate of movement, and it is hard to maintain uninterrupted connection. Those make multiplayer mobile applications more challenging.

In recent years, the uses of mobile devices become more and more popular. Most people have mobile devices, even while some people have two or more mobile devices. On the other hand, more and more applications for mobile devices have been developed. Using the mobile device to handle personal affairs is a part of modern life. The mobile application is one of the essential software for new generation of mobile devices.

However, the operation of the mobile environment has been a lot of restrictions, including:

1. The limited computing power of mobile devices:

The complicated operation is not suitable for use on mobile devices; making computing time cannot be applied too long.

2. The limited storage space for mobile devices:

The memory of mobile devices is generally much smaller than the personal computer, cannot save or share a lot of space for large files.

3. Connectivity for mobile devices will be affected by the movement:

Mobile devices, by definition, is used when we move, it is difficult to guarantee that there will be a stable environment where the network connection. For example, move to the basement, tunnels, elevators and other confined space, will make the connection lost, cannot access network resources.

4. The limited interface for mobile devices:

The input interface of mobile devices is usually little more than the average number of personal computers, it is difficult to support fast keyboard input and mouse operations. On the other hand, the output screen of mobile devices is also much smaller than the personal computer. We cannot put too much information in the same screen at the same time because of too small to read. Therefore,

applications for mobile devices must be adjusted to meet the user easy to use.

1.2 Motivation

Today, mobile application platform can basically be divided into two types:

One is to develop products for its own system only; application is not portable to other platforms, such as: Android market [8], I-phone App [9], Windows Mobile Widgets [10] ... Another is to provide a common platform for applications development, such as: Yahoo! GO [11]. Yahoo! Go provides the JavaScript [12] with XML [13], Widget written [14], and for different mobile devices to provide a different execution environment for Widget platform.

Widget is one kind of web applications and it is compact and easy to develop using standard web technologies such as: HTML, CSS and JavaScript, so most of the program developers have the ability to develop such applications. Therefore, more and more developers join the rank of Widget development.

Yahoo! Go is able to support many mobile platforms, because Yahoo must spend a lot of time and invest heavily in technical resources to overcome many compatibility problems to develop for different platforms in different execution environment. However, this design concept will make the platform must be accompanied the transaction of environment to do regular updates and corrections, even substantial changes. For this reason, I thought maybe we can build a Browser-based platform. It can reduce the overhead for the developer to correct the platform in order to be suitable for varies device.

The Browser-base Widget engine can also be divided into two categories, one is

implementation of client-side, and the other is the implementation of server-side. The benefit implemented by client-side is that we don't need a server to be required to provide Widgets, but this kind of engine cannot be transferred to the other browsers because of the different core of browser. In contrast, the benefits of server-side widget engine is easy to cross-platform and also can run in different browsers, the disadvantage is that this engine can only be used on-line, when off-line status, they become useless.

1.3 Research Objectives

This thesis will use the existing standard mobile browser as execution environment, and develop a Cross-platform & Cross-browser execution environment for mobile Widget applications. Use the browser as an interface; developer can perform the same Web applications on different platforms. Besides, we will choose a server-side widget engine to let this environment can work in the different mobile browser. Moreover, we will find a mechanism that allows applications to operate offline. Finally, we will design a user-friendly interface which provides users controlling our system easily.

Basically, the current type of widget can be divided into three types, the first type is the information provided by the remote server, such as: weather forecasts, RSS news, business information, etc. The second type is the content will be stored some records in database, such as: To-do list, Notepad, etc. The third type of Widget, the main part is composed by other services, such as: Dictionary services, Search services, On-line Map, etc.

Because of XML tag of Google Widget is supported by a considerable number, and

even some grammar can be used in the other Google services such as Google maps. So many grammars, it is difficult to be completed in this thesis, therefore, we expect to select some important and basic grammar to support.

In this thesis, we will make a To-Do List and a RSS Widget to show how our system works and how the program developers let their Widget to support off-line functionality.

The pictures as follows:

1. To-Do List

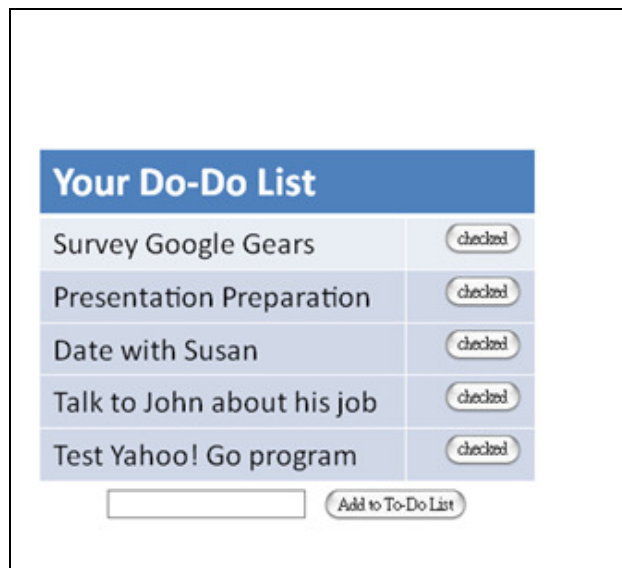


Figure 1-1 To-Do List

As above, the users record what they want to do through this To-Do List on mobile devices. And, if the user has added a new item to this To-Do List in off-line state, the data will first buffer to the local database of this mobile device. Waiting for users to re-connect to the network, it starts to do data synchronization.

2. RSS Widget

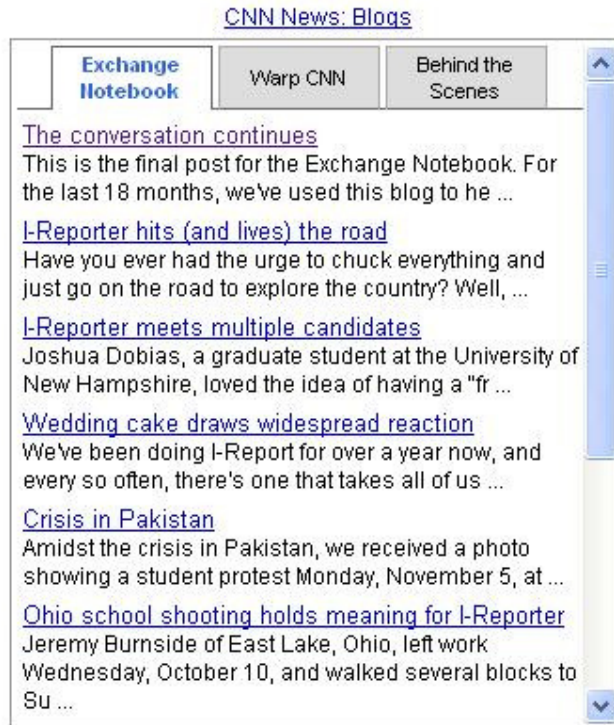


Figure 1-1 RSS type of Widget

As the above diagram, this is a simple RSS [16] style widget, through this widget, users can use the simple way to read news or events for each outline, if they press more interested in the subject, they can also through the hyperlinks into the original page to read it. After the implementation of our system, we hope to achieve that anyone can read the news list under each tab in off-line state. We also hope that we can spend the least space to read the original news to achieve the effect of offline reading.

1.4 Research Contribution

In order to achieve the objectives mentioned in section 1.3, we have encountered many difficulties during implementation. This thesis discusses those problems encountered and our corresponding solutions. The major contributions of this research

are listed below:

1. The developers of Widget engine do not need to solve the compatibility problem.
2. The Widget programmer can easily let their application work on different environment and does not to take care of off-line operation.

1.5 Outline of the Thesis

This dissertation is divided into six chapters. Following is a brief description of the content of each chapter:

1. In Chapter 2, the background of developing this system and the related work including some commercial products are introduced.
2. In Chapter 3, an overview of the proposed system and its major components are given.
3. In Chapter 4, the implementation details, problems encountered, and our corresponding solutions are illustrated.
4. In Chapter 5, using practical examples show the widgets work in our system and compare our system with the other platform.
5. In Chapter 6, we dwell on conclusion and future work for referencing.

Chapter 2 Background

This chapter will give you the background of our research. And we give an introduction on other related work including several commercial products and academic efforts on this topic.

2.1 Google Gadget

2.1.1 Introduction of Google Gadget

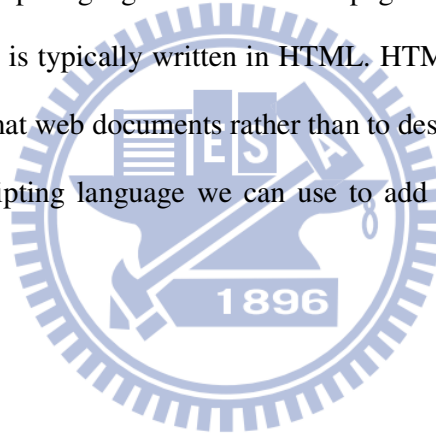
Google Gadget and Yahoo Widget [17] are also the desktop Widgets. They are using JavaScript write small programs to facilitate interesting features. Google Gadgets are dynamic web content [18] that can be games, mini applications, news feeds, maps or any content. Members of Google's site can add them to their iGoogle page. With Google Gadgets anyone who has a Google account can create and publish them. Other users can add the Gadget to their iGoogle page or alternatively web designers can copy and modify the Gadget and put it on their web site.

Currently Google Gadget can be used in conjunction with other Google API to create more diversified small tools, such as: Google Maps API, Google AJAX Search API, Google Calendar API, and Google Translate API. As long as include the library of those API in the code that we can successfully use these services in the gadget.

2.1.2 Main APIs of Google Gadget

The gadget APIs consist of a few simple building blocks: XML, HTML, and JavaScript.

- XML is a general purpose markup language. It describes structured data in a way that both humans and computers can read and write. We will write the XML file to specify a gadget contains instructions on how to process and render the gadget. The XML file can contain all of the data and code for the gadget, or it can have references for where to find the rest of the elements.
- HTML is the markup language used to format pages on the internet. The static content of a gadget is typically written in HTML. HTML looks similar to XML, but it's used to format web documents rather than to describe structured data.
- JavaScript is a scripting language we can use to add dynamic behavior to our gadgets.



2.1.3 A sample of Google Gadget

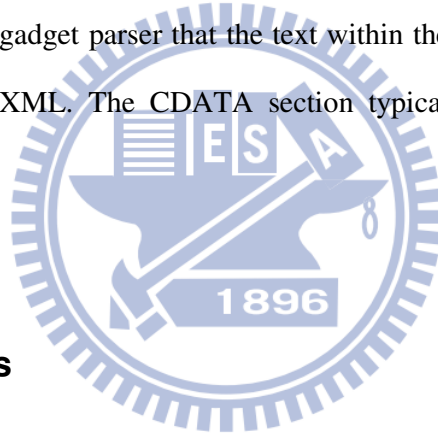
The simplest gadget is just a few lines of code. This gadget displays the message “Hello, world!”

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="hello world example" />
  <Content type="html">
    <![CDATA[
      Hello, world!
    ]]>
  </Content>
</Module>
```

Figure 2-1 Sample of Google Gadget

Note the following about the "Hello World" example:

- Because Google Gadgets are specified in XML. The first line is the standard way to start an XML file.
- The <Module> tag indicates that this XML file contains a gadget.
- The <ModulePrefs> section in the XML file specifies characteristics of the gadget, such as title, author, preferred sizing, and other optional features. The users of this gadget cannot change these attributes.
- The <Content type="html"> indicates that the gadget's content type is HTML.
- The <![CDATA[...]]> is used to enclose HTML when a gadget's content type is html. It tells the gadget parser that the text within the CDATA section should not be treated as XML. The CDATA section typically contains HTML and JavaScript.



2.2 Google Gears

2.2.1 Introduction of Google Gears

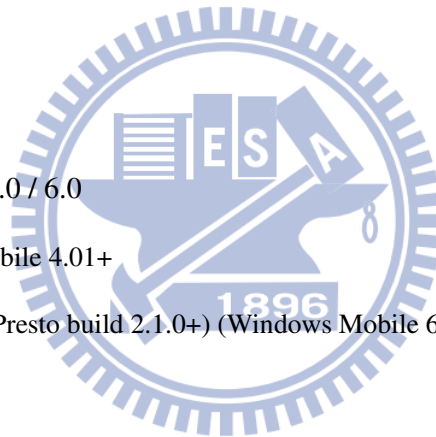
Google Gears is an open-source browser extension. It supports for off-line web applications, can be used in desktop computers or handheld devices. Applications can obtain resources from the local storage that we pre-fetch it. It can be used for full text search in a database. It uses asynchronous JavaScript [12] operations to help improve responsiveness of application.

Google Gears API contains a variety of different components, so that web browsing has more new features. The more significant elements of Gears such as for example:

- LocalServer: Provide a local server cache and application resources (In HTML, JavaScript, images mainly)
- Database: Database stores data locally in a fully-searchable relational database
- WorkerPool: Low resource consumption of asynchronous operation, maintain the interactive response of Web Application

Google Gears can be used in desktop computers or handheld devices on different operating systems. As follows:

- Windows XP/Vista
- Mac OS/X
- Linux
- Windows Mobile 5.0 / 6.0
- Internet Explorer Mobile 4.01+
- Opera Mobile 9.51 (Presto build 2.1.0+) (Windows Mobile 6 touch screen only)



2.2.2 System architecture and off-line theory of Google Gears

Google Gears for development purposes is how to make web applications to reach off-line browsing. However, to promote off-line feature, there are always some issues that must be considered:

- I. How to set the data layer
- II. Decide when to take online or offline, or connection strategy
- III. Select how applications present in on-line or off-line state
- IV. How to construct the synchronization functionality

We illustrate in accordance with the above four issues as following:

I. How to set the data layer

The traditional web applications, users should be through the application programming interface to send requests directly with the remote server. There is no data layer between the two settings:

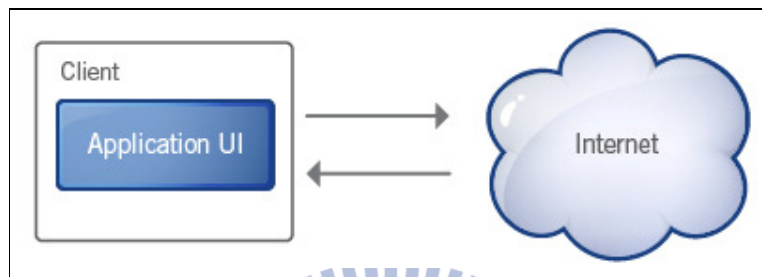


Figure 2-1 Traditional requests

Google Gears join the data layer on the local side; it is an access point to store information what applications request for. So the remote data will have a place to store on the local side. And then, the application interface will obtain information sources from remote capture, change to obtain the data layer. As shown below:

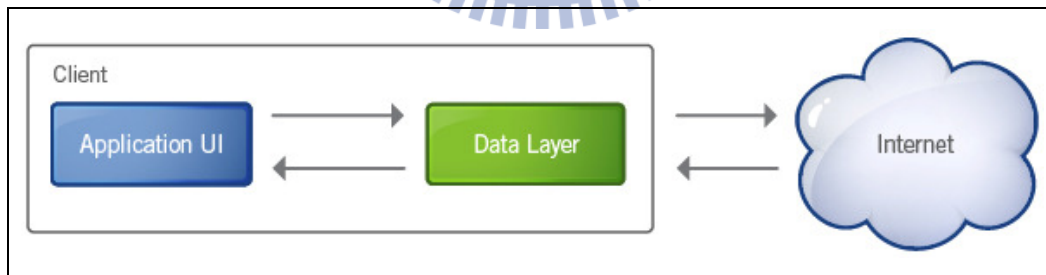


Figure 2-3 Join the data layer

Then add data switch layer, used to determine the source of pages of information is from an data layer (locally),or directly from the Internet (external) to obtain, it still can both use; when the information of data layer needs update, this layer can decide when to write data locally, or sent to Server for synchronization. As shown below:

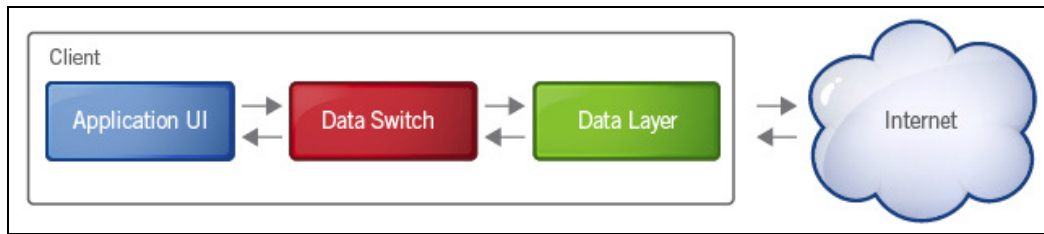


Figure 2-4 Data Switch layer

Google Gears use another way to create a new local data layer; the layer uses a Gears database instead of going to the web server for data. In the off-line case, data switch layer request information for the Local Data Layer, and then Local Data Layer get webpage data from the Google Gears database:

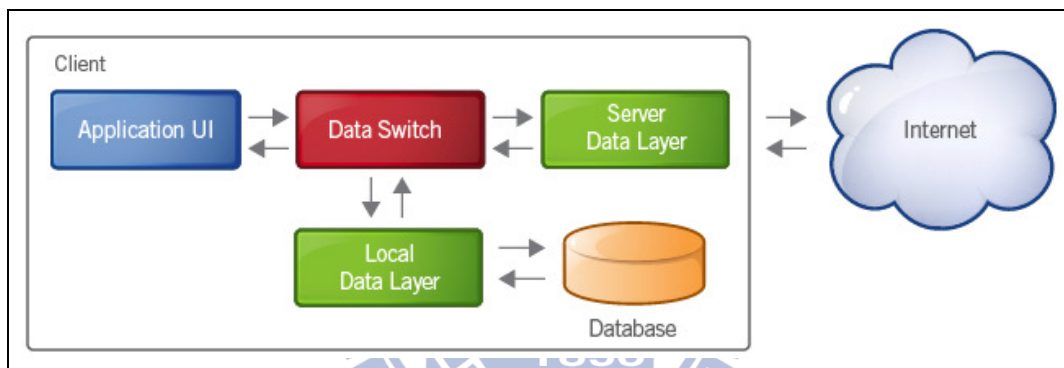


Figure 2-5 Database of Google Gears

II. Decide when to take on-line or off-line, or connection strategy

Off-line web technology need to consider when to use local information or replace the server directly connected at different timing, known as the "connection strategy". Although the better efficiency is that access information from the local, but actually, a local server, not all suitable to replace the information provided by remote server.

For example:

- Source server for the real-time stock quote system
- Server contains huge amount of information

Based on the speed of local side is faster than the remote, so the better approach is still possible store by the local side. In comparison, the greater amount of information stored on the need to synchronize the greater amount of information.

III. Select how applications present in on-line or off-line state

There are two modes that applications are displayed or not in off-line state.

The two kinds of differences:

- The application displays off-line mode:

There will be some differences between on-line and off-line state. Sometimes users need to switch the mode themselves.

Such as: Google Reader: It will connect with server in on-line state or use local offline storage. Information will be synchronized each time when the users switch. This model is relatively easier to implement. Drawback is that users need to remember to do mode switching, or sudden off-line will be no information available; if the network quality is unstable, users need to get alternative use for the model.

- The application does not display off-line mode: There will not be any significant difference. The user does not need to switch connection mode, the program will operate automatically.

Such as: GearPad: It is destined for off-line execution environment, so at any time if we lose network connectivity will not be affected. This model allows users to ignore the state of mode switching network, even in the unstable network can continue to use as usual. Drawback is not easy to implement, we need to avoid consuming too many resources for background synchronization,

then decrease the overall performance. Synchronous operation has nothing to do with user activity, so testing difficult.

IV. How to construct the synchronization functionality

Usually, through the operation of program, the contents of the local and server will be different, when the connection of network works, the contents need for synchronization; synchronous operation mode can be used manually or background. However, Google Gears does not provide synchronization engine, which must implement by the application developer according to their own development.

■ Manual synchronization

As the manual synchronization is not at definite time, so every time they must pay attention to the problem of load limits. In addition, users may not understand the situation on the network, only find the network is not available when they try to synchronize, or forgot to synchronize before they enter an environment which network is not available. Manual synchronization with easy to implement, but require user participation features.

■ Background synchronization

In a network available environment, the application can be continued synchronization between the local side and server, and detect the network connection before doing synchronization. In such way, if a sudden failure or instant off-line, we still can maintain the latest information.

Below is the Google Gears chart includes background synchronization.

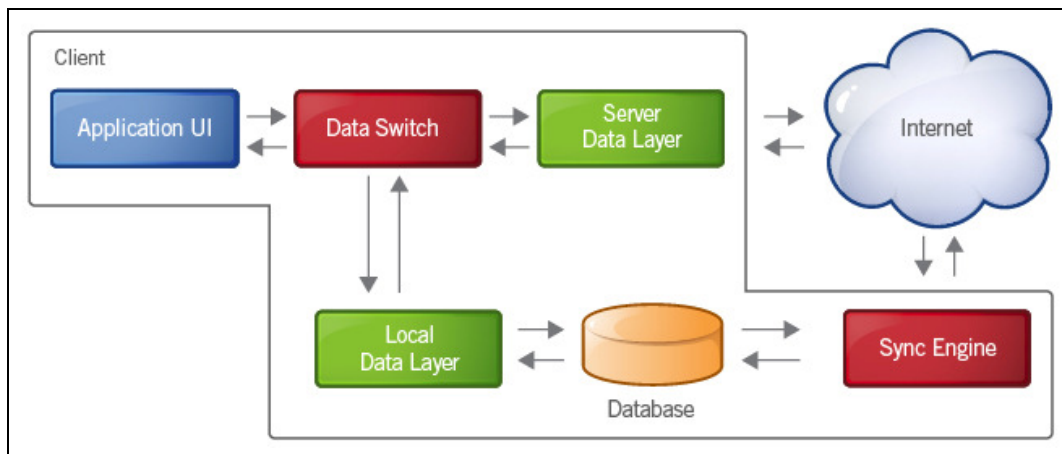


Figure 2-6 Sync Engine of Google Gears

2.2.3 The information security of Google Gears

Google Gears [15] uses the Same Origin Policy [19], which specifies the source of information on the local side for access. In order to protect users, each site for the first time produce off-line information before, the warning dialog will be presented, every web sites is essential that users permission to the implementation of Google Gears. All the local information will differentiate their accounts; it cannot be used interchangeably in order to protect personal privacy.

2.2.4 The functions of Google Gears API

The functions of Google Gears API all are summarized as follows:

- Factory: The Factory class is used to instantiate all other Gears objects.
- Database module: It provides browser-local relational data storage for web application...
- HttpRequest module: Let the WorkerPool have ability of HTTP request.

- LocalServer module: It allows a web application to cache and serve its HTTP resources locally, without a network connection.
- Timer module: It's a timer to provide time records for the use of other API.
- WorkerPool module: This API allows web applications to run JavaScript code in the background, without blocking the main page's script execution.

2.3 Apache Shindig

2.3.1 What is Apache Shindig?

Apache Shindig is a server-side-based Web application platform and can be rendered Widget XML into html code.

Apache Shindig provides two languages (Java, PHP) reference implementation in this project; we constructed a PHP version of the implementation.

Shindig itself is divided into four main parts:

1. Gadget Container JavaScript:
Required for general gadget JavaScript, and manage security, communication, UI layout, and feature extensions.
2. Gadget Rendering Server:
It is used to render Gadget XML format into JavaScript and Html.
3. OpenSocial Container JavaScript: We don't use it in this thesis.
4. OpenSocial Data Server: We don't use it in this thesis.

2.3.2 Five different servlets in Apache Shindig

Shindig contains 5 different servlets, which are described below:

Each servlet is the subclass of HttpServlet, and serve as independent entry points at runtime.

1. GadgetDataServlet:

It takes requests for loading social data from file into DataObjects.

It takes requests for OpenSocial APIs.

2. GadgetRenderingServlet:

It converts Gadget Xml into HTML.

In this thesis, we must do some fix in this part.

3. Proxy Servlet:

It provides URL Gadgets to connect to the remote URL, and fetches the contents then turns into JSON format and sends it back.

In this thesis, we must understand how the Proxy Servlet fetches the remote contents and then make some amendments for our system.

4. JsServlet:

It is used for Loading JavaScript libraries from external sources in URL gadgets.

5. RpcServlet

It is used to handle RPC metadata requests.

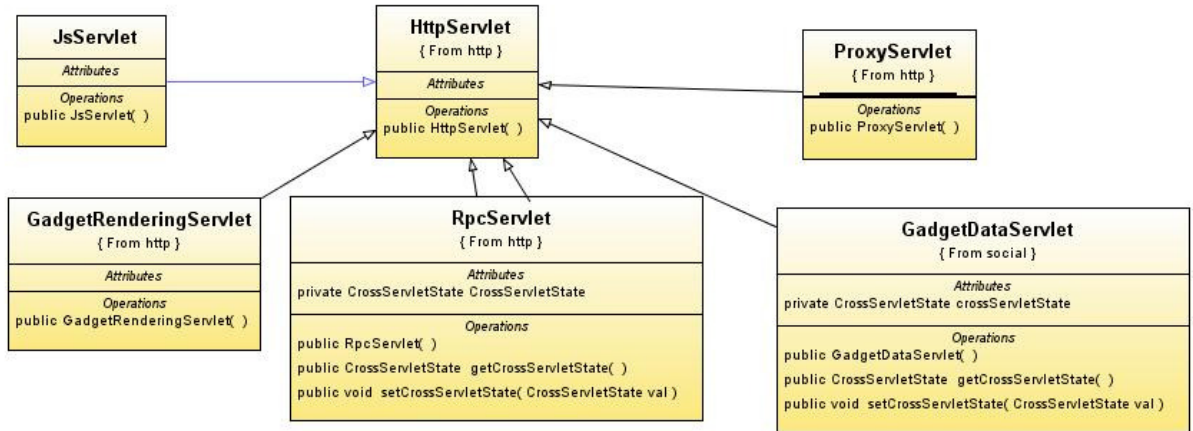


Figure 2-7 The Class diagrams of these five servlets

2.4 Related Works

In order to understand the other's platform with our differences and know what we should take care when we implement such a platform.

The following sections we will introduce several other related applications platform. And also provides Html 5 [20] introduction to understand how we use the next generation browser and the possibility of making web application

2.4.1 Yahoo! Go

When an Internet company wants to provide mobile services, he must face a problem that is a large number of mobile phone operating system, and different brands of cell phones have their specific functions, or different browser set in numerous mobile phones. In order to solve this problem, Yahoo! Go offers an environment (a Java-based phone application) for the current site can be simple and easy for operators

to put their services on mobile phones, and not worry about what underlying technology used in the background. But to achieve this goal, Yahoo! may own a lot of compatibility issues to overcome. Actually, Yahoo! announced in late December 2009 that Yahoo! Go along with technical support for it would be discontinued on January 12, 2010. About January 22nd, Yahoo! Go no longer works.

Yahoo! Go aims to their application in each mobile phone can run normally, it must be compatible with the operating system of various mobile phones. He must be for different mobile phones or browsers to be adjusted to ensure that the function is normal and display successfully. As long as the phone supports Yahoo mobile developer platform, website operators just follow the development of specifications in accordance with written Widget; it will be able to provide existing web services for mobile phone users who have installed Yahoo! Go software.



Figure 2-8 Yahoo! Go software

Yahoo! GO has a lot of services covered the basic news browse, read mail, weather forecasts, even to view the latest Flickr photos. Other features include Yahoo search engine, we can easily search for other sites, as long as input the site name, for

example, enter: 01, mobile search will bring Mobile01 website link, click on the link, we can immediately visit the site; Other built-in features include Yahoo News, entertainment, finance, weather, sports, it would be a RSS receive service, not to endure lengthy downloads or start connections, we can update at any time the latest current affairs of life.

Yahoo! Go uses Widget for the conceptual design. Widget is designed to immediately access the site resources. Furthermore, taking into account the small window, it is designed to let user reading comfortable. The developers can use Yahoo! Widget tool produces self-developed Widget. Yahoo! Widget is a technology using a JavaScript open source platform, to support Windows and MacOS X operating system. Yahoo Widget tool provides thousands of desktop mini-applications, these tools in the Yahoo Widget platform to run mini-applications, called the Widget tool. By these Widget tools, users can link to Yahoo's web services to obtain all the required personal information.

Yahoo! Go developed to version 3.0, open advertiser embedded advertising, and to provide external developers to use Yahoo's Widgets functionality, design more diverse content for mobile users, so that external developers can put into the keyword advertising to create revenue.

2.4.2 Web Page Tailoring Tool

As mobile devices become more popular, the opportunity to use them to go to the Internet becomes more and more. However, the vast majority of pages are designed for the computer. When using the mobile device browsing these pages, because the

screen size limit, the user might always have to scroll in very inconvenient way. Although some websites specifically for mobile devices provide an additional mobile version, but through this approach, Web developers should spend more time to design and maintain the same sites; In addition, some commercial web pages that will let original content of the page to re-layout, for example: let Web into long strips in order to remove the scroll trouble. However, the re-layout the web pages may not be able to allow users to quickly find the desired information, because still there is some unimportant information.

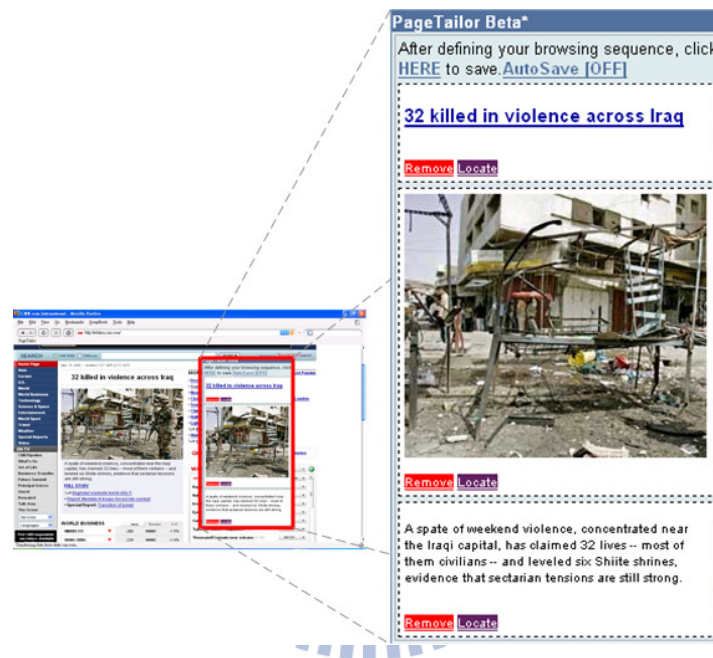


Figure 2-9: Page Tailor in Web browser

In view of this, Chi-Yang Tsa from National Chiao Tung University [21] creates a web system to help users to get what portion they want to read. The system can determine Web pages according to users' personal preferences. And it facilitates the mobile user to read the small screen.

The main principle is to allow users to use JavaScript to select what part they want to read. As the Dom tree [22] of html can be expressed as an internal way, so this tool will be transform the areas user selected into a Dom tree path, and save to the database.

When users actually access, they will set up a Muffin server [23] of this system as a proxy server. This server is a World Wide Web filtering system written entirely in Java that can filter any HTTP data sent and received by user's Web browser. This system is rewritten muffin server and according to the Dom tree path which is saved in the database as a basis for filter data. Users only set a range once and the other pages in the same path of the web can be made by the same set to do filter.

2.4.3 HTML 5

HTML 5 is a new network standard, aiming to replace the existing HTML 4.01, XHTML 1.0 and DOM Level 2 HTML standards. It hopes to reduce the need for browser plug-in-based rich internet application, such as Adobe Flash, Microsoft Silverlight [24], and Sun JavaFX [25] needs.

HTML 5 provides some new elements and attributes to reflect modern usage of the typical Web site. Some of them are technically similar `<div>` and `` label, but has some meaning, such as `<nav>` (navigation block) and `<footer>`. The tags will help search engines to index order, small screen devices and the visually impaired. It provides new functionality for other new requirements, such as `<audio>` and `<video>` tag. Some HTML 4 tags will be removed, including the tags used solely for display, such as `` and `<center>`, because they have been replaced by CSS.

Currently Firefox, Google Chrome, Opera and Safari (version 4 and above) has supported HTML5 technology.

Here is a list of the APIs of HTML 5:

- The Contacts API
- Selectors API Level 1/2
- Programmable HTTP Caching and Serving
- Indexed Database API
- Web Workers
- Web Storage
- Web SQL Database
- Server-Sent Events
- XMLHttpRequest level 1/2
- File API
- Geolocation API Specification
- HTML Canvas 2D Context
- HTML Microdata
- Capture API
- Notification API

Here a few of API in the future will replace the Google Gears, so we illustrate with them:

■ **Web SQL Database:**

As Google Gear, the browser will has a local database functionality, which provides a local SQLite database as the core, this way, people in the Client-side can have a simple SQL database for information, so let the front page to access it.

■ **Web Storage:**

A Name-Value-based data storage area used for HTML & JavaScript, the information stored there will continue to be a local database file, even turn off the Browser will not disappear, it can be used as a storage area similar Cookie.

Chapter 3 System Design

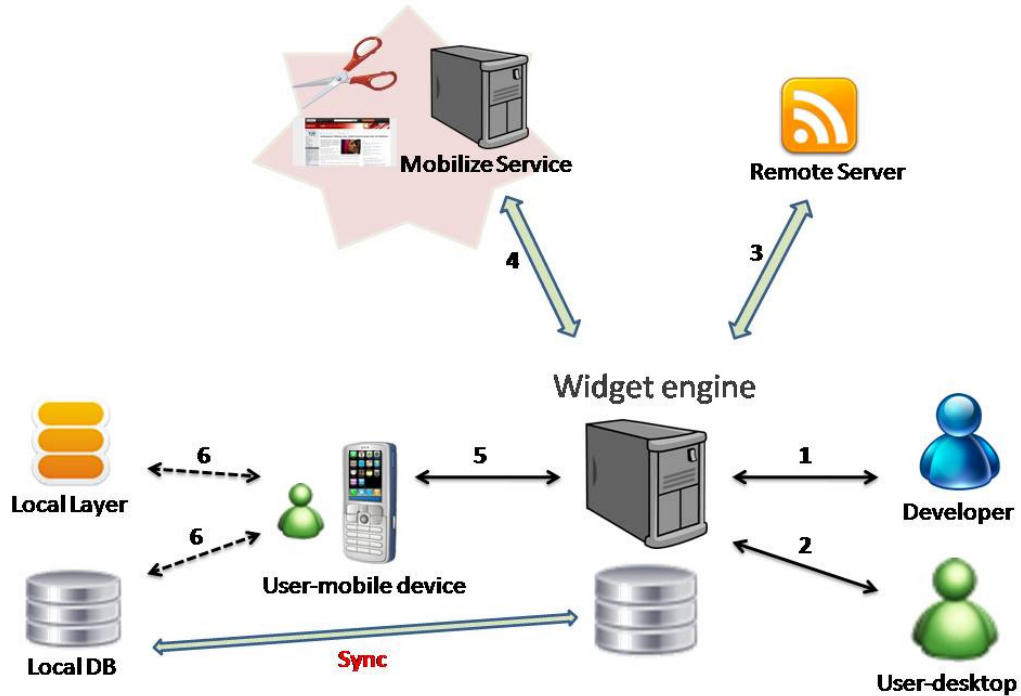


Figure 3-1: Overview

3.1 Overview

At first, Application Developers write Widget according to their needs. Because we use Apache Shindig which is an Open Source to be modified as a Mobile Desktop Server, so as long as follow the standard Google Widget rules, the widget they write can be easily used in our system.

Application Developer may put the complete picture which is required for the development of the Widget and RSS list xml or the website information on their Application Server.

Because Google Widget can contain JavaScript syntax, so the application

functionality required to conduct and support Google Gears are able to be included in the form of JavaScript in the Widget XML file.

This system process is as follows:

Step1 :

When the developer completed development of its own widget, then they can through the entrance page we set up to upload the icon which is representative of this widget and the widget xml into our server.

In addition, if the widget produced by developer will get the information through the remote server side that we can use tailor tool to set the information provided by remote server, in order to let remote server's Web content more suitable for the mobile platform.

Step2 :

In this step, user can through our entrance pages to select want the widget they want, including each user and their selected widget will be recorded in the our server's database (this database created and designed by us).

Step3 :

Our Widget engine will be obtained in accordance with developer upload widget xml, parse each of which tab is based on remote xml, and he crawled down (in order to meet the gears in the mechanism), then to rewrite the contents inside.

Step4 :

The Widget engine adapted by us will let the remote xml crawled by step3 to do an extra parse, and according to the link found after parse, send these http-request for

each one. After the response received, and we will go through a proxy server to do extra tailor treatment

Step5 :

Under the environment of the mobile user when the selection has been completed, as long as the first connected to the our server can get their widget

Step6 :

In the first through to our server to get users' own widgets, Server will keep the content of html required by off-line state in the Gears of the local layer, so that the future for off-line use. In addition to the first access, the rest of the time only when the content of html changes, then we just verify the content of local layer

Sync :

If the developer's widget designed to record some of the specific items needed (ex: To-do List), it will use the database belongs to Gears to record the transactions of the machine, and when on-line state, it will be through the internal system to achieve the backend synchronization.

3.2 System Architecture

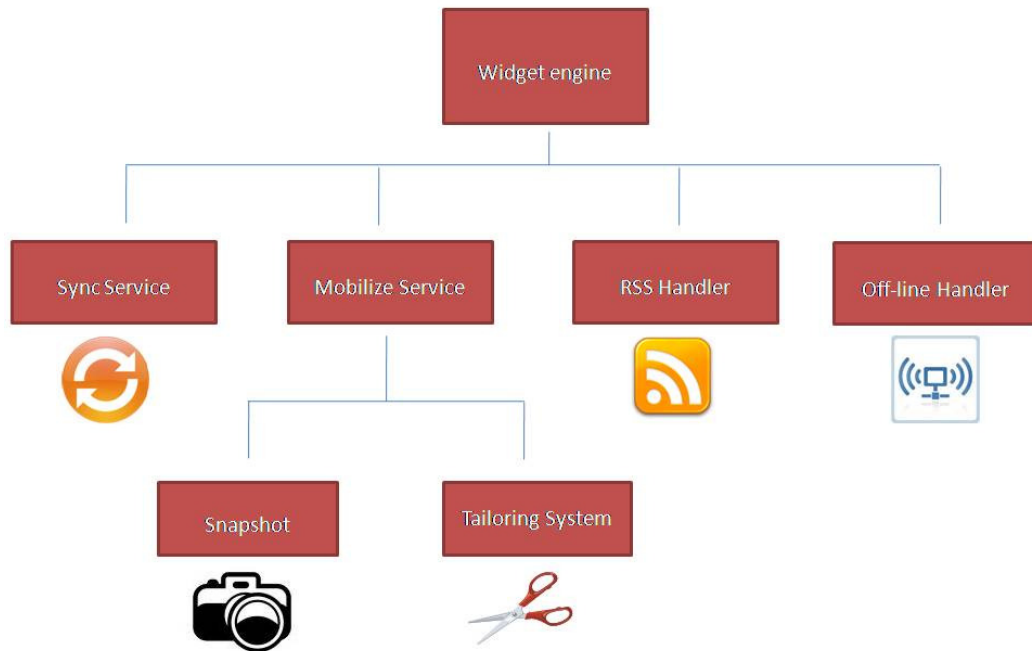


Figure 3-1 System Architecture

Within a single view on our system can be divided into four component, these are the Sync Service, Mobilize Service, RSS Handler, Off-line Handler.

Below we elaborate on the four components to be:

- **Widget engine:** Correctly render the content from the widget to the html results.
 - **Off-line handler:** Let the Widget application can be used off-line to become a general application.
 - **Mobilize Service:** Let web pages be mobilized, and reduce the burden of mobile space for off-line read.
 - **Sync Service:** In the online state, synchronize the server-side data and mobile data, maintain the mobility feature in our system.
 - **RSS Handler:** Let our server-side Widget engine can be able to follow Same Origin Policy, and let data has multiple meaning.

3.2.1 Mobilize Service



Figure 3-5 Display of Mobilize

The Figure 3-5 show the Tailored Webpage after mobilized by our service.

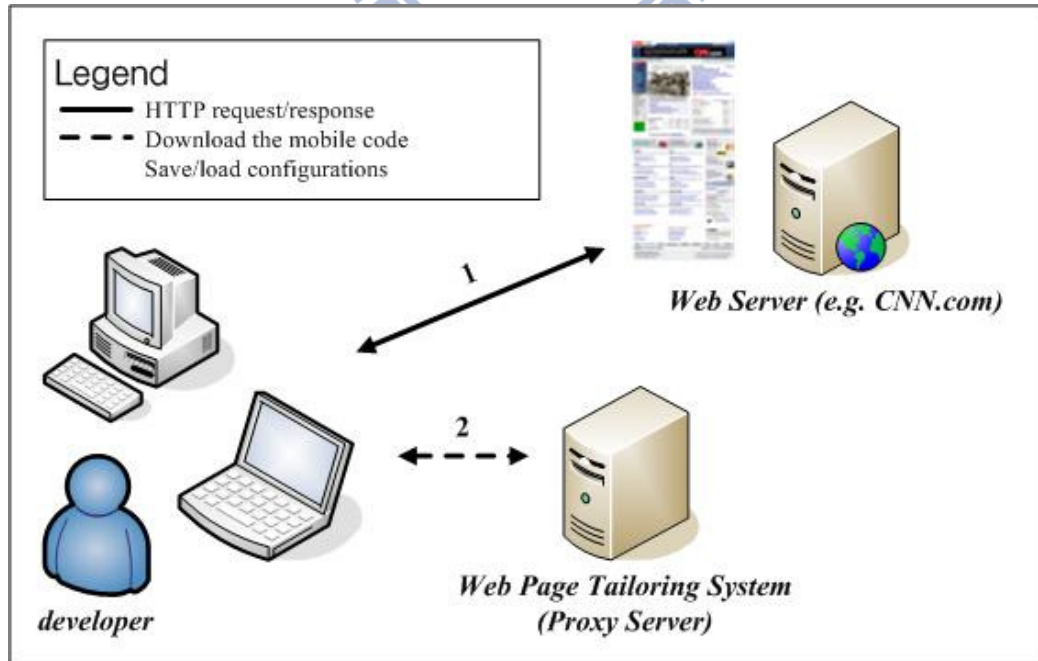


Figure 3-3 Web Page Tailoring System

Developer can use the desktop browser to go to their own web pages, and then select the tailor part for mobile.

Carried out as follows:

1. Loading external JavaScript Libraries
2. Select the main part of the supply of mobile browser viewing
3. Write what they select into Webpage Tailoring system

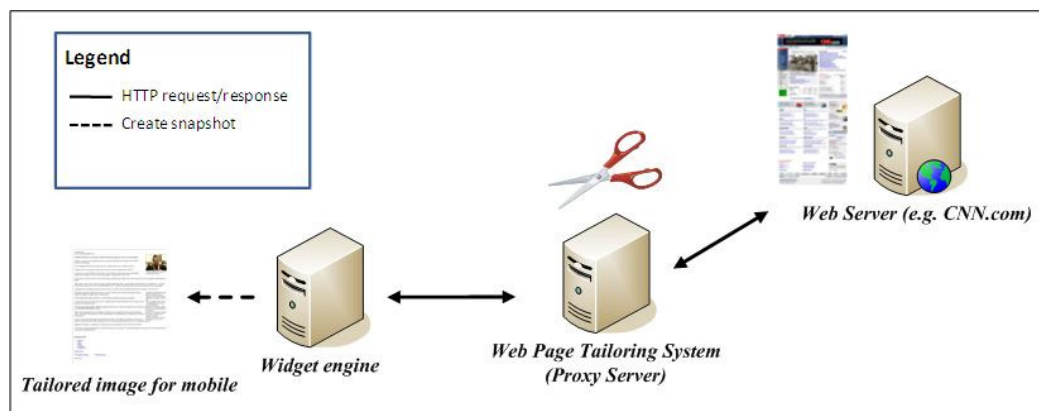


Figure 3-4 Combined with our system

The Widget engine which is rewritten by our system will be in accordance with the original content from the RSS List xml to get link, and then one by one through the webpage Tailoring System to collect information from Web Server.

It can be seen from the figure the original page data, if this domain has been set before, then post the information after Tailoring System will become streamlined many.

As Widget engine where the browser we have already set it will be collecting information through a Proxy Server, so the resulting content with Tailoring System will be seen within the same, also, we use a SiteShoter program to save Web page data which the browser render out as jpeg format for offline use.

3.2.2 RSS Handler/Parsing Widget

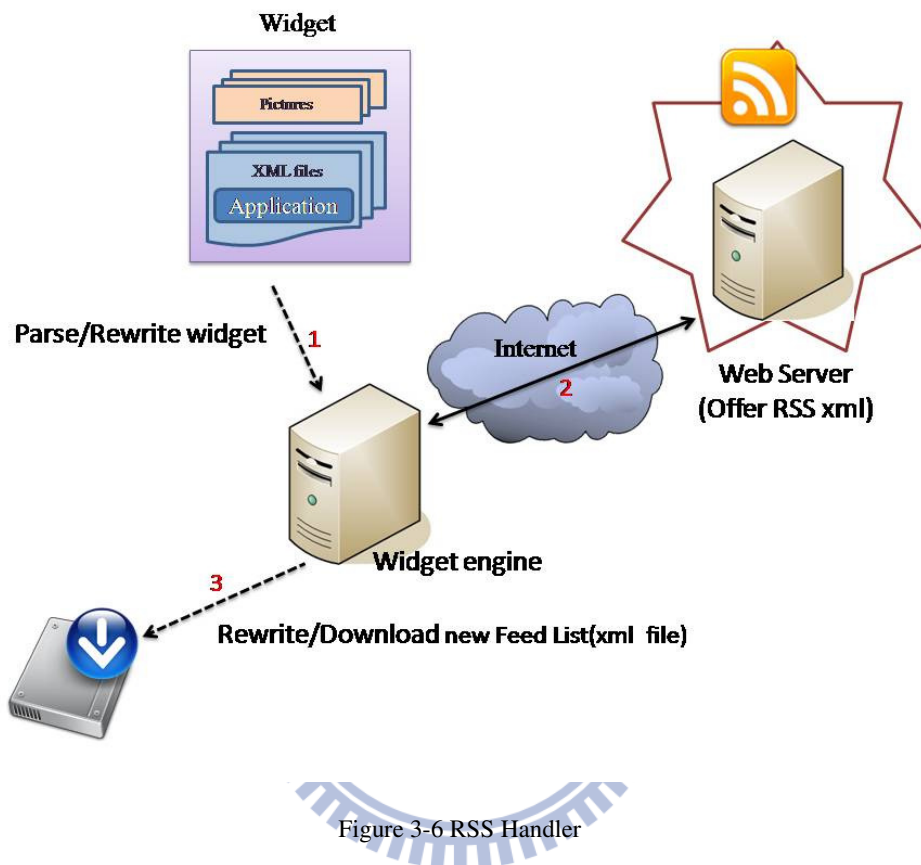


Figure 3-6 RSS Handler

After developer upload the widget which is completed development, Widget engine will begin to parse this widget. Our system will rewrite each sub-page's link and points to our own domain, and then one by one through the Internet send request to their Web Server in order to obtain the corresponding RSS List xml. Basically the RSS List xml is one type of material within the widget.

In order to allow Gears to access, we must download this RSS List xml.

In addition, for the purpose of dealing with subsequent Tailor tool, the various sub-projects in RSS List xml will also has been changed by server to be saved into the image file format of a link.

3.2.3 Sync Service

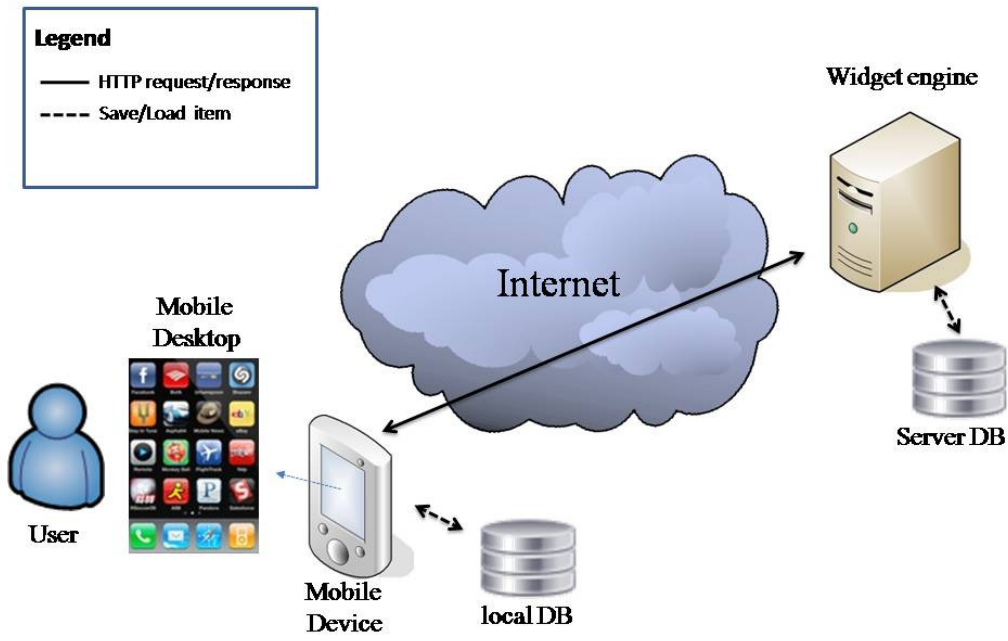


Figure 3-7 Sync Service

Each time, when user on line and use the To-do list that the type of widget will use the database. We include the JavaScript code in the widget will periodically call synchronization function, as the synchronization function is accomplished by the Ajax, all the action on or synchronization are automatically in the background.

3.3 The behavior of Mobile device

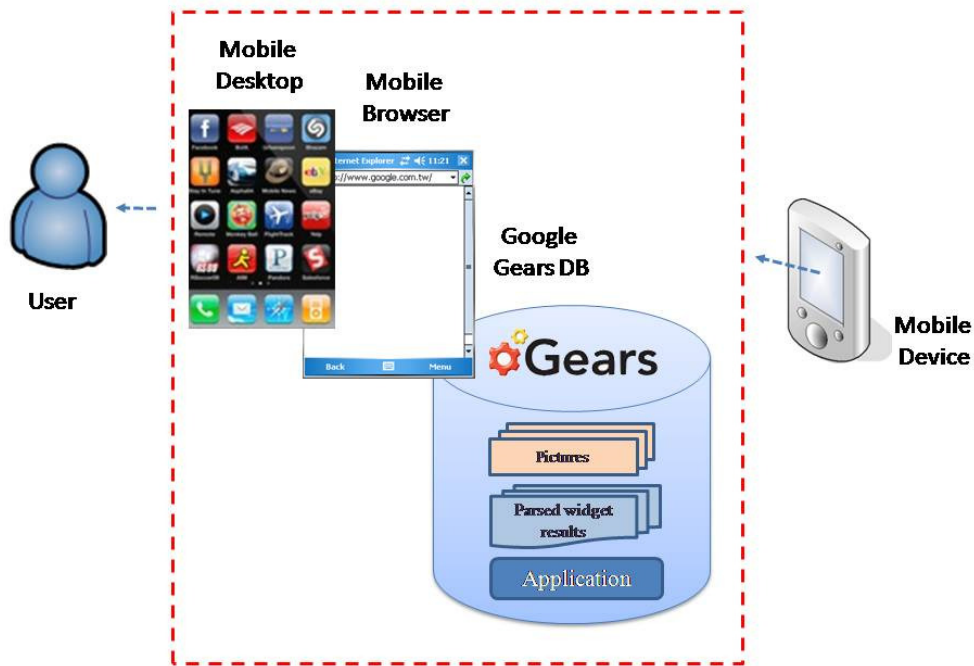


Figure 3-8 The behavior of mobile device

Before the off-line state, the user as long as download the registered Widget and all related files then stored in Google Gears. It can be maintained in the off-line as the normal use of these applications after connection lost.

The only difference is that, if the types of widget such as To-do List widget will stop synchronization in the offline with the Server-side; and Feed type of widget change the original request which is sent to Application server into the other request Gears offer and go to their local layer to obtain information.

3.4 Summary

In the beginning of this chapter, we introduce how our system works. A developer uploads their Widget using PC or laptop and then through our system, users can easily

download the widget and the other related off-line data into their mobile device. In addition, this system implements a browser-based offline web application environment for Widget,

The main components are:

1. Use Widget engine by we rewrite to render widget xml into html & JavaScript.
2. Using RSS handler to let our server-side Widget engine can be able to follow Same Origin Policy, and let data has Multiple meaning
3. Using off-line handler to let the Widget application can be used offline to become a general application.
4. Implement sync handler to let the Widget application which has database can be synchronized with our server.
5. Using mobilize service for web pages be mobilized, and reduce the burden of mobile space for off-line reading.

Next, we describe the major components in our system: RSS Handler, Mobilize Service, and Sync Service. The design concepts and functions provided by each component are detailed in separate sections.

Chapter 4 System Implementation

In this chapter, we will describe the implementation details of the major components in our system by sequence diagram and some figures. And introduce what problems we encountered during implementation. Of course, our solutions to these problems are also depicted.

4.1 Application Development

4.1.1 Sequence Diagram of Application Development

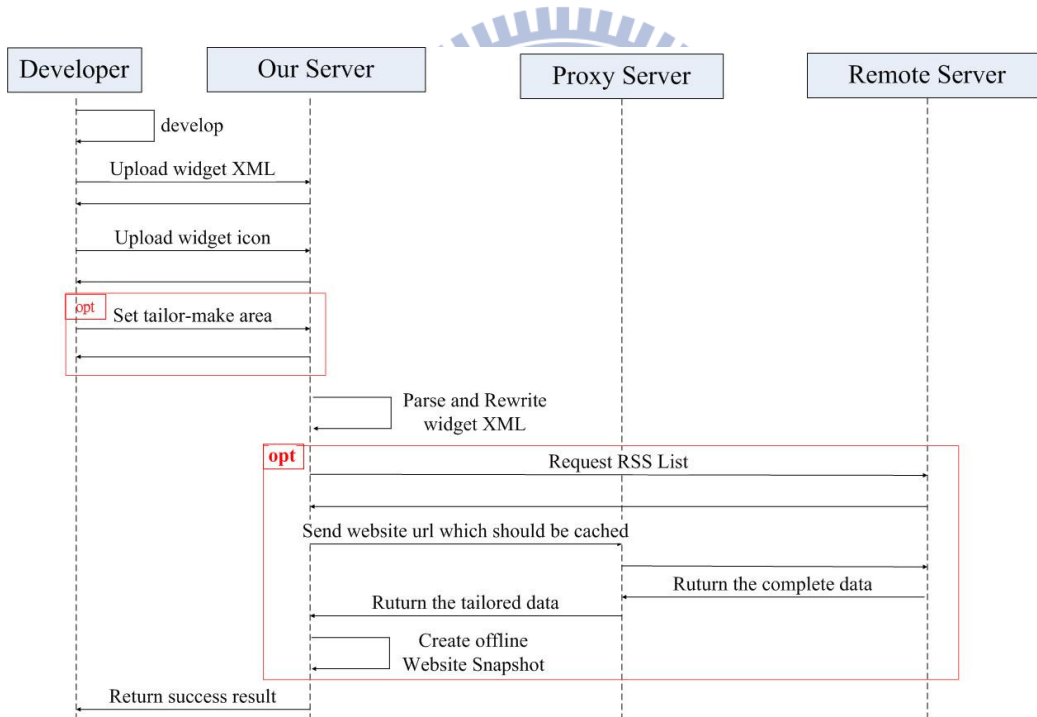


Figure 4-1 Sequence Diagram of Application Development

As shown in Figure 4-1, at first, developer built their applications according to the standard Google Widget. And he made the application icon, as users click on the basis of the phone. After completing its development Widget XML and icon images, then upload to our Application Server. If this widget is belong to RSS type then the tailor

tool can be used to tailor the area to be set by developer. Later, our Server will parse the widget xml developer uploaded. The reason that we parse it is in order to identify what type of widget written. And then our system will grab the remote xml as that reference the contents of the Widget xml. In addition, the content of RSS widget can be changed to be able to operate with Gears in the form of what we rewriting. This is because Google Gears downloaded file must be in the same domain of Server. Therefore, once our Server received the registration requirements of the application, our system will automatically be related to XML and image files, and then download it to the Our Server. The reason for this is to facilitate in the future for the Google Gears in mobile phone to download. In addition, if the type of widget is RSS, in order to be able to read offline links which is correlative with various actual content of RSS News, we will do the second parse for the RSS xml.

After we got all the News Link, we will send requests to the Remote Web Server. At first, Remote Server will return the complete page. However, through the proxy server which is established by us, the page will be modified in the form for mobile use.

Finally, it can be achieved by a Snapshot program from a complete page to turn into the image file jpeg format.

Completed, we will notify developer “Upload successful” message.

4.1.2 Implement details of Rewriting Widget

Next, we began to discuss the real problems we encounter. Firstly, because the operation of off-line mode is to use Google gears mechanism. The Google gears itself in accordance with the Json [27] file to access the list of information contents. However, due to security issues, all items of the list in Json file content must be the same domain with program server which is provided this Json file. For this purpose,

we must copy all the information which we need to read it in off-line state to our own server. For this reason we must analyze what information is generated by the developer will be using in the widget, in this thesis we study the most two basic type of widgets written by developer. In the first type of widget, that is one kind of RSS news, developer sets widget content is provided by these external links, this is why we have to resolve where the external links the widget's position in. After the real contents of these external links were dumped to our server, it just completes the first step of the parse work. After crawl down the external RSS list xml, we will conduct a second parse. However, part of the discussion with the SiteShoter, we will explain it later.

■ How to add the local links into cache list

```
1 {
2   "betaManifestVersion": 1,
3   "version": "version 1.0",
4   "entries": [
5     { "url": "go_offline.html"},
6     { "url": "go_offline.js"},
7     { "url": "http://140.113.88.199/new_xml/test.xml"},
8     { "url": "http://140.113.88.199/img_txt/BBCNews_89.txt"},
9     { "url": "gears_init.js"}
10  ]
11 }
```

Figure 4-1 Cache list

The "entries" attribute contains an array of URLs that we wish to capture and have available offline. For every file and image that we wish to capture, add a "url" entry into the "entries" array. Follow this format we show it like the above sample manifest file. In this place we will fill the real link of local server which we covert it from remote server in the Json file.

4.2 User Subscription

4.2.1 Sequence Diagram of User Subscription

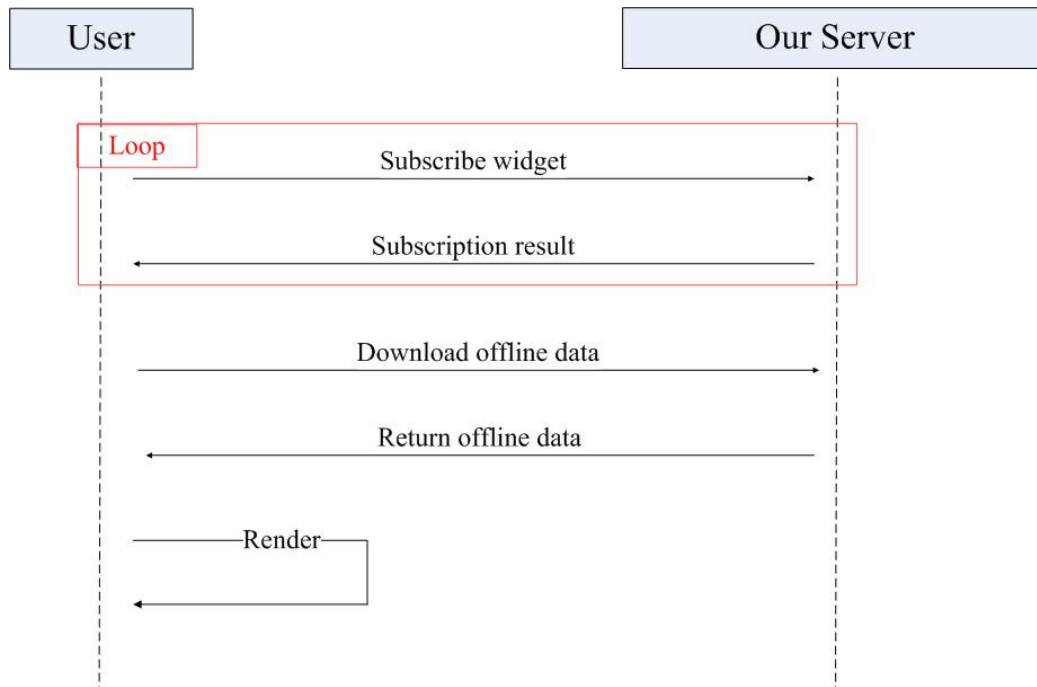


Figure 4-3 Sequence Diagram of User Subscription

Shown in Figure 4-3, if the user's mobile device maintain in on-line state, users will be provided through the Server page, select what he want to subscribe for the widget. After successful subscription, Server will return a successful subscription message to notify the user. Thereafter, the user then subscribe to all the relevant files Widget, by way of Google Gears to download. (At this point the page file will includes the main webpage and related image file) Since the required application logic can be included in the pages written in JavaScript, so when the user is off-line, he can use Widget stored in Google Gears and the relevant application files for offline operation.

4.3 Implementation of the two demonstrations

4.3.1 Sequence Diagram of To-Do List

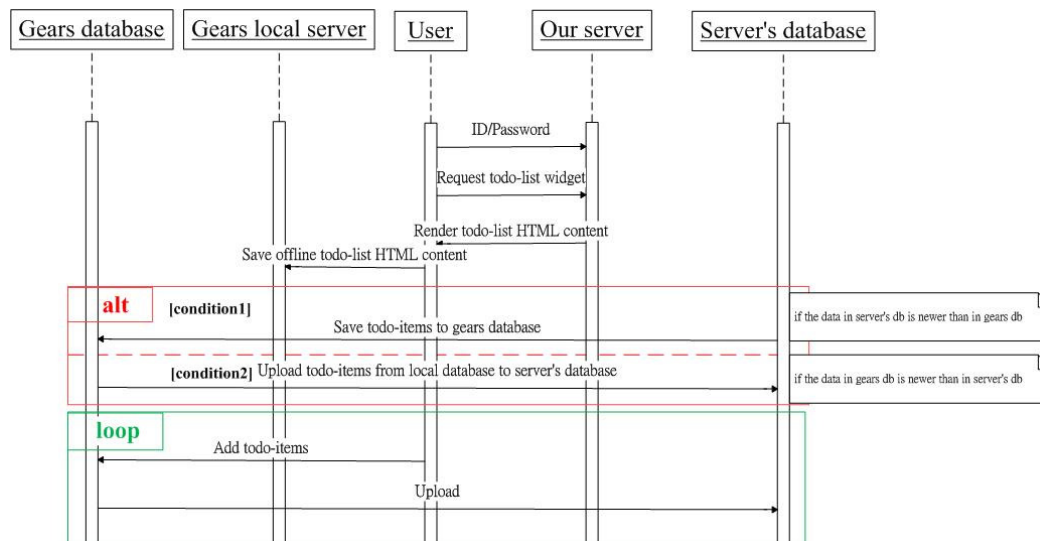


Figure 4-4 Sequence Diagram of on-line To-Do List

As shown in Figure 4-4, if the user's mobile device maintain in on-line state, users enter the account number and password to login our server. Sign in to our server to obtain To-Do-list of the HTML content and then store it in the Gears local layer for offline access. If the version of information stored in server's database is newer than stored in the Gears database, the system will automatically update to the new data into the Gears database; contrary information in Gears database will be uploaded to server's database. Also in the To-Do-list when users add a new item, the system will write information on the local end of Gears database firstly and automatically uploaded to the server's database for synchronized action.

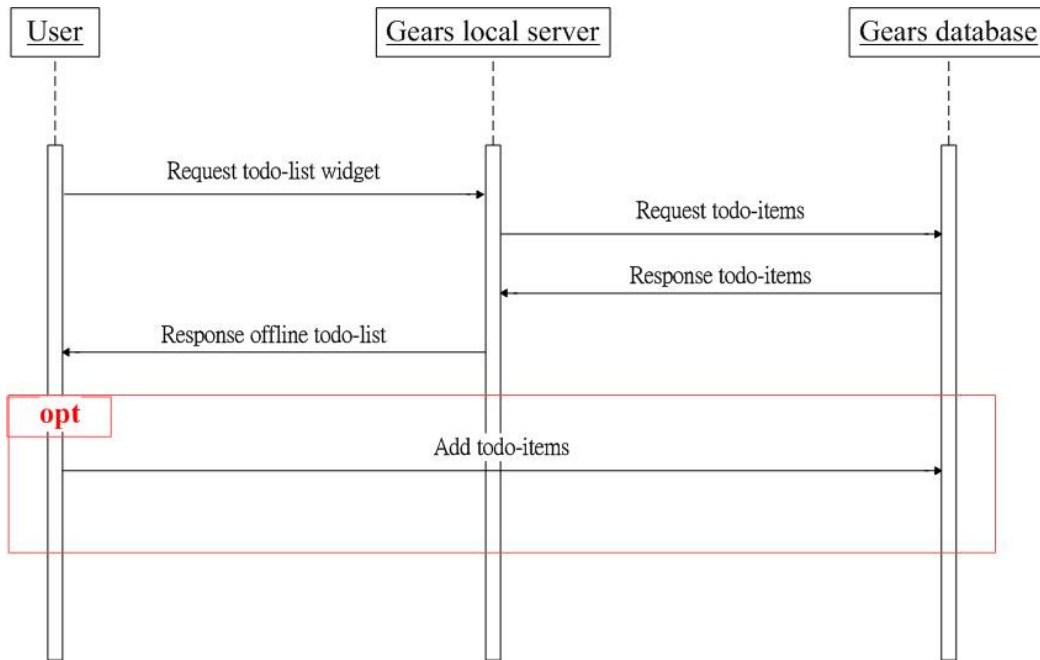


Figure 4-5 Sequence Diagram of off-line To-Do List

As shown in Figure 4-5, the user's mobile device offline, when users want to use the To-do list gadget, the user will be through the local server to the Gears database to obtain information, and then back passed to the user. If users add the To-do list new items in off-line state, the system will be in the first write to the Gears database, until the connection work then do synchronized movements.

4.3.2 Implementation of Sync Service

```

98
99 □ gadgets.Prefs.prototype.set = function(key, value) {
100     var needUpdate = false;
101
102     var need_store=value;
103
104     Insert_local_db(need_store);
  
```

Figure 4-6 Prefs function

```

49
50     function saveItems(){
51         prefs.set("items", JSON.stringify(items));
52         _gel("newItemInput").focus();
53     }

```

Figure 4-7 Implementation of Save/Load

The origin Shindig server provided the function “Prefs” to allow developer to write information on the storage and how to handle the materials, etc., but it did not set “Prefs” function of actual behavior, so we have to write server-side custom storage. In addition, client's behavior, we must be written within the server side, including within the JavaScript with the transmission to client. As part of sync is to be behind the deal, so our front-end is to use AJAX framework to complete it. Besides, because the actual program writing, and the convenience of comparison, we keep all items which is present by Json format together to enter into the database, not only keep a single 1 items in such a mechanism in place, in this way, we only compare the serial number of ID that we know what we write at first. Another point is that, since the purpose of this thesis is that we can use widget smoothly in the offline state, so all available information will be stored firstly in the phone side and then go to complete the sync action.

■ **Do developers can write the sync function by themselves?**

Another point in the sync part needs to be mentioned, that is, if the developer can write their own sync methods or not? After writing the original widget and developer offer their widget to igoole platform, they do not need sync function, because all the stored data will be kept to the Google server. But the way we structure, data must be stored in the server side and the other local side. However, this behavior of store will go through their library from the server to actually execute, and we rewrite the library from the server side to complete the sync correction features. So even if the

developers to customize the other sync function, it is unable to change the library. Therefore, that conclusion is developer cannot be completed to do the sync action without our help.

4.3.3 Sequence Diagram of RSS Widget

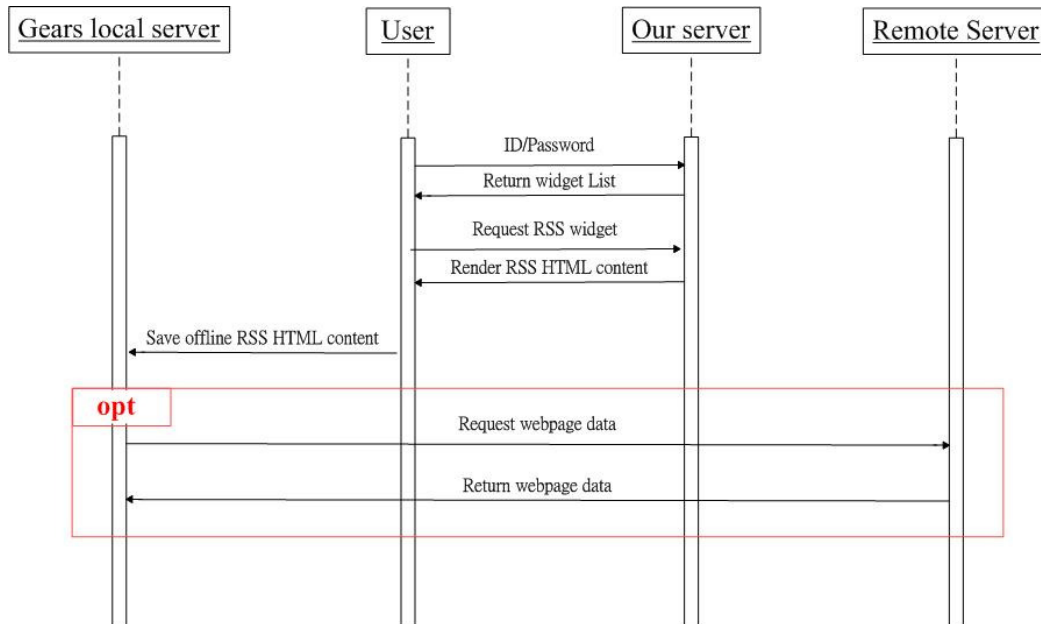


Figure 4-8 Sequence Diagram of on-line RSS Widget

If the user wants to use this type of RSS widget xml, it's nothing different with the above To-do list widget. Log in first through our server, and get selected in the widget, and when the first transmission or the information content of transaction, we put the html results which is rendered out by Server into Gears. If in the on-line state, we want to connect to external resources, the way it just like that we usually use a normal Webpage, request for the external resources, and then receive information.

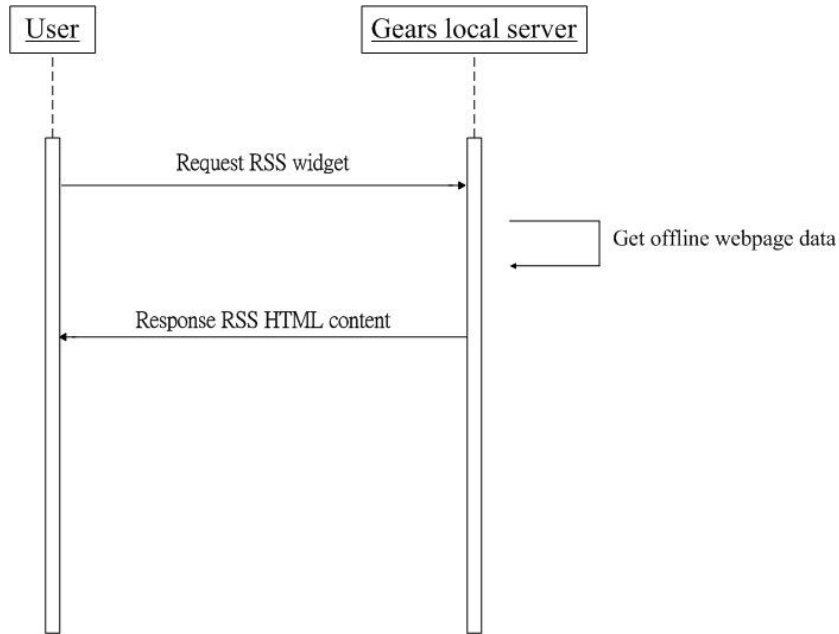


Figure 4-9 Sequence Diagram of off-line RSS Widget

If this type of widget is in off-line state, so basically, all requests will be shifted to gears, the gears of the local layer to provide all necessary information.

```

18 <item>
19   <title>Israel to free flotilla activists</title>
20   <description>Israel says it will release and deport foreigners seized on boar
21   <link>http://news.bbc.co.uk/go/rss/-/1/hi/world/middle_east/10210949.stm
22   @http://140.113.88.199/url_pic/BBCNews_97_0.jpg</link>
23   <guid isPermaLink="false">http://news.bbc.co.uk/1/hi/world/middle_east/102109
24   <pubDate>Wed, 02 Jun 2010 00:53:25 GMT</pubDate>
25   <category>Middle East</category>
26   <media:thumbnail width="66" height="49" url="http://newsimg.bbc.co.uk/media/i:
27 </item>
  
```

Figure 4-10 Add off-line link

In order to allow user collect information from local server in off-line state. We add the local link corresponding with the original link in the back of original link. So the client-side can easily obtain information from the same place when context switching between the two mode.

In addition, it may encounter some difficulties or something needs to be changed in

implementing this idea.

■ **Replace the XMLHttpRequest function by the similar library of Google Gears**

When the on-line state, we can use XMLHttpRequest to get the webpage data from the remote server. In order to meet off-line, we will be changed to use gears to provide XMLHttpRequest information to the local side. (Because we cannot know the actual location to store the offline data in gears, so if we want to get the offline data to handle, it only such this way can only be used) But according to the actual situation to the presumption, gears do not complete some access. For example: offline Send XMLHttpRequest to fetch xml data, if the information is too large, then it will appear errors.

So we condense to such a result to do transaction with the format of our data storage.

■ **Some JavaScript is not supported in off-line state**

In addition to different JavaScript behavior on different browsers, nearly completion of the program also had another very troublesome problem. In the offline state, some JavaScript is not supported. Since the behavior of certain programs we hope to be able to handle at the back-end. For example, we must judge off-line and on-line or not right now. At first, we use Ajax architecture to implement this behavior. And we deal with the results then set it to be synchronously responded.

Such as the following way :

```
XMLHttpRequestObject.open ("GET", url, false);
```

```
//The flag "false" means it must be synchronized.
```

Originally in the on-line state, operation as they were set, it is very successful. But once the change is set to off-line status, it just finds the program cannot successfully work. Later it was found that mobile browser does not support off-line Ajax.

Therefore, the original handling of Ajax must be changed to do a similar deal. We just can use the library of gears to do so. But the library of gears has provided inadequate. For example, we can use the flag to set synchronization response in the originally `HttpRequest.open` function. However, there is no such flag in gears. Therefore, the original structure has to be rewritten with do such acts.

4.4 The other problems we encountered

■ The version of Apache Shindig Server

As the Shindig is an open source, so when we developed this service, at the same time Shindig is still doing amendment. At first time, when we finish a written program, it sometimes cannot be used after we update the version of Shindig server. Then understand why, we put the program fixed in the present version. The current version we have taken tags-1.1-BETA2. However, after testing, there are still some widget has been developed in accordance with the Google widget format cannot work successfully in the Shindig server. But, if we keep the structure and direction of this thesis to do an amendment, basically, we will not need to fix it too much after the final version appears.

■ Cache issues of implementation

According to the framework adopted in this thesis and the use of language, the most common problems when we are writing are cache issues. Browser will always automatically cache website data. Ajax request made using a cache of the issue will arise; Shindig server will be temporary space issue. Offline operation of gears also needs to update the version of Json file, and then that would be to download the new transaction. These problems are after repeated testing to know to understand. Basically, it does no problem in our architecture, under these conclusions really be

difficult to debug.

The following sets out the various cache-related solutions:

1. Browser cache problem
 - A. From the Safety menu in the upper right, click Delete Browsing History
 - B. Deselect Preserve Favorites website data, and select Temporary Internet files, Cookies, and History.
 - C. Click Delete.
2. Using Ajax to send request, the cache of issue have produced
Set Header : `request.setRequestHeader ("If-Modified-Since", "0");`
3. Shindig server have temporary space of issue
Clear “tmp” folder generated by Shindig server side in the local machine.
4. Update the contents of gears in off-line state
Change the json file which is called tutorial_manifest.json. And then, update the number of this sentence "Version": "version 1.4".

■ **No a good debug tool of mobile browser**

In addition to cache problems, we use JavaScript to implement the action what the Shindig server do. Because there is no comprehensive debug tools. In many cases, only some clerical error, no complier (similar to c or java) will tell you where the error occurred. We just find out a little bug will spend a lot of time. Combined according to different browser, the JavaScript support varies. Sometimes we use the same operation can be worked on IE, but using chrome appear the errors. Even on the table browser support, the operation transferred to the mobile terminal does not work sometimes. The browser on mobile environment even does not have debug tools can be used (such as Firefox’s firebug [28]). This again hinders the development.

4.5 Implementation of caching Remote WebPages

■ The reason to use SiteShoter

Originally, we want to use the structure of Tung-Hing Chow's thesis [29] to cache the real webpage content which is pointed by RSS link to present in offline state. But, in fact, when we do so, we found that because the kind of structure that is to use Htrack to complete the original data crawl. Unfortunately, there are many server provided RSS news; they use a rewrite module practical guide to re-link position. (It is a redirect rule).

Ex:http://www.cnn.com/exchange/blogs/notebook/2008/03/conversation-continues.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed:+rss/cnn_exchange+notebook+ (Blog: +Exchange+Notebook)

Therefore, when we use Htrack this program, we cannot fully grasp the corresponding information and files what we should grasp.

(Because the architecture of Htrack is to capture the information which is upper or lower with the relative path)

	Before mobilize /hierarchical	After mobilize (save as a jpeg)
Single remote webpage	0.8 mb	0.25 mb
Completely Widget	20 mb	6.25 mb

Table 4-1 Comparison table of Mobilize effect

In addition, we may have to spend just one page a few kilobytes of space to access the required images or information. A widget is likely to be five or six themes, if there

will be five RSS news for each theme. Then there will be 25 pages about the information. In order to grasp it down after the developers upload their widget, it is equivalent to $25 * 0.8 \text{ mb} = 20 \text{ mb}$ of space to maintain a widget. And in this way, then user must wait for a very long time to download the widget available offline, and also waste short of storage space for mobile phones. So we try another way to complete the offline Web page => Store the content as image format in advance.

The beginning we wanted to integrate this application with the our server and implement by PHP language. However, after we actually write it, it shows that we can complete the basic snapshot. But we do not know how to store a bar web page to a picture. Maybe we can take the time to continue the implement, and then we can complete it, but it deviate from our original direction. So we look for a program to implement what we need, and the program is Siteshoter.

■ **Implementation of Permission issue**

In addition, the implement of this part still encounters a problem. Siteshoter will open a browser in the background, and visit the designated website (that is the received parameters of the program). But if when we calls this Siteshoter program on the web server environment, Operating System will treat it as a request from the web service. In this case the inherited permissions cannot use another tailor services together. So we have also set up a java server to call this program. Let its property belonging to his application level. Then, another website tailor tool can be operated normally.

■ **Set Time issue of SiteShoter**

After the second parse, our server will write the links of webpage that we want to use Siteshoter to crawl in the same batch. So in the next update, they can deal with actions

in one step.

But, processing in Screenshoter also encountered a problem. Since each web page will use a new browser to handle. But we must set a time to do the crawl action. If at that time point when the browser is not completely capture the content of webpage down. Then the content is going to be wrong. Maybe the time point can be postponed, but the time of each page we crawl it will be prolonged, then the overall waiting time will become longer.

4.6 Summary

This chapter includes all the implementation details of the major components that together compose our Browser-based web application platform. The implementation issues and their corresponding solutions are addressed here, too



Chapter 5 System Demonstration & Evaluation

In this chapter, we will present a demonstration for our system. In this demonstration, we will describe a general scenario with a handheld device of Android which is installed “Google Gears” already in its browser. Moreover, there will be an evaluation of the comparison between our system and related works at the end of this chapter.

5.1 Basic Operation

5.1.1 Developer uploads widgets

Step 1: Go to the developer page: `http://<Server IP>/portal/index_developer.php`

Step 2-1: Specify the file name to upload files and select the xml file, thumbnail and the applicable browser then press the **【upload】** button.



Figure 5-1 Developer uploads the widget file

Step 2-2: After a successful upload, the file will be shown below



Figure 5-1 Uploaded

5.1.2 User selects widgets

Step 1: Enter the user login page, click on **【new user】** to register new user.

Website: [http:// <Server IP> / portal / login.php](http://<Server IP> / portal / login.php)

Step 2: Register for a new account

Step 3: Back to the login page, enter the registered account, and the password.

Step 4: Enter the user home page, click the icon on the left side management interface.

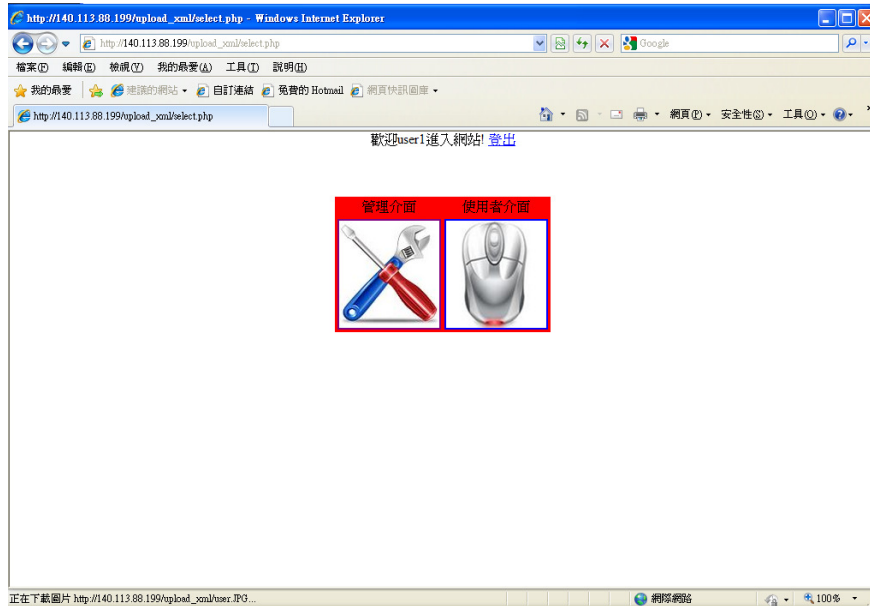


Figure 5-3 The user's Homepage

Step 5: Enter the management interface, the existing file will be displayed in the bottom, and the top will show the selected files.

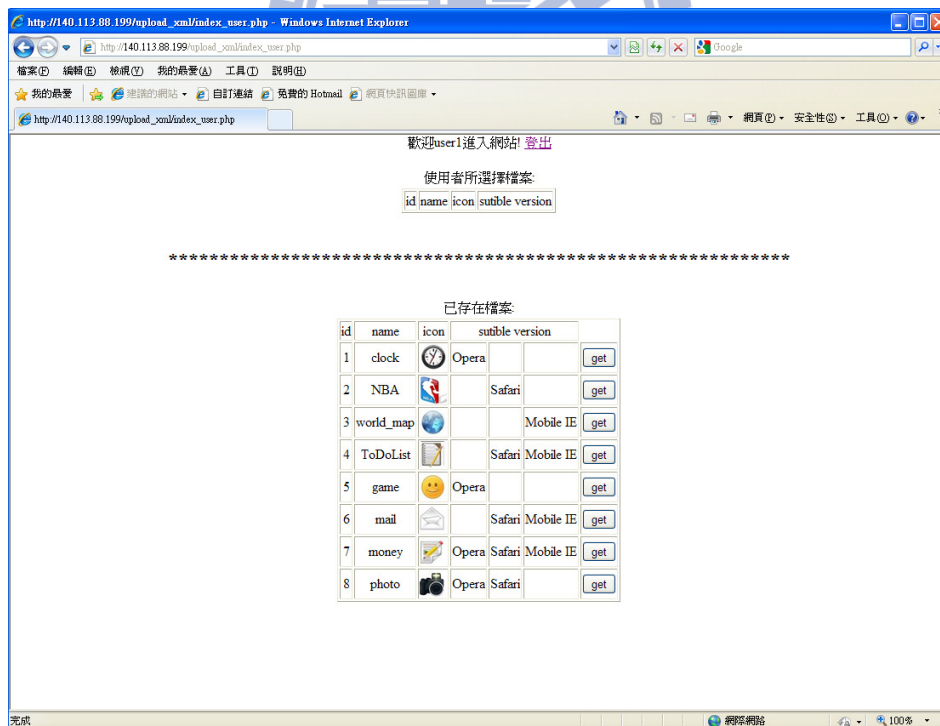


Figure 5-4 User Management Interface

Step 6: Users can select the widget they want to use, here we select To-Do-List as an example, click the **【get】** button, the user's file list will add this widget. Users can also click the **【remove】** button, the widget from the list will be removed.

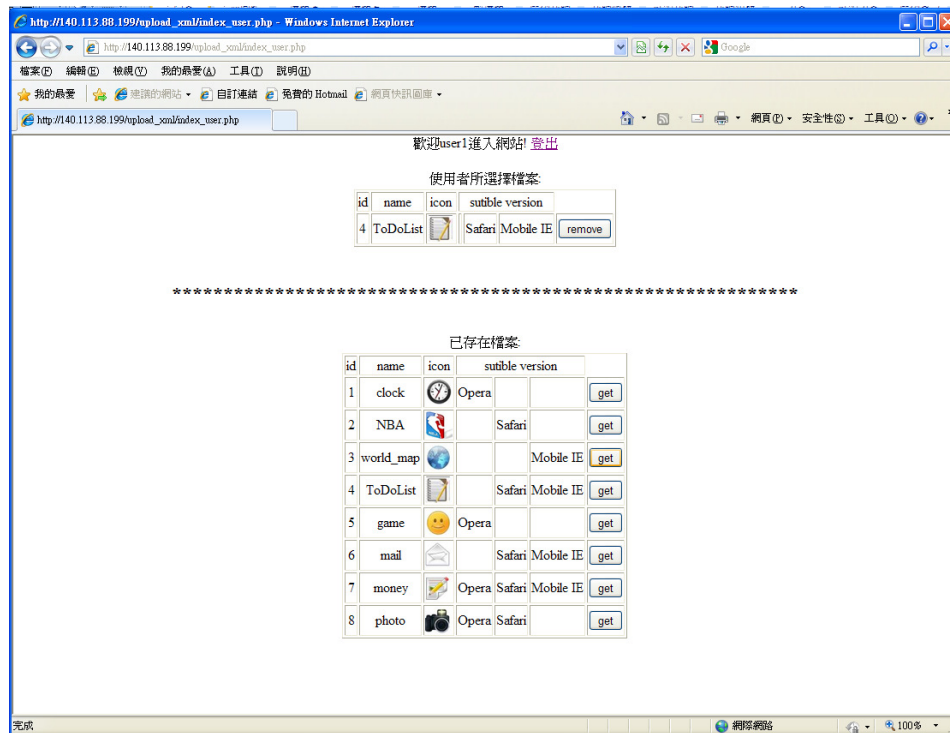


Figure 5-5 The user's selection completed

5.1.3 Do operation in mobile phone

Step 1: Switch to mobile user's login page, enter the registered account, password.

Website: `http:// <Server IP> / portal / login.php`

Step 2: Go to the user's home page, click the icon on the right side of the user interface.



Figure 5-6 Users' Homepage in mobile device

Step 3: Users can click on any icon to use the widget.

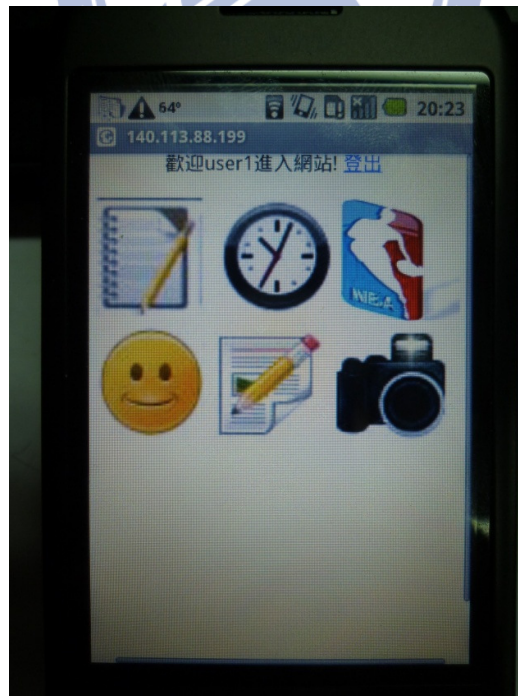


Figure 5-7 Users' Widget Interface

5.2 The Demo of To-Do List

5.2.1 Functional test and Synchronization test in on-line state

Step 1: From the user's interface, click on To-Do-List icon, it will appear Google Gears prompt message, select "I trust this site", press the lower right corner **【allow】**, so we can enable offline functionality.



Figure 5-8 User allows Google Gears

Step 2: Enter To-Do-List Widget page, in the top field, enter the new item we want to matter, right click the **【Add】** button.

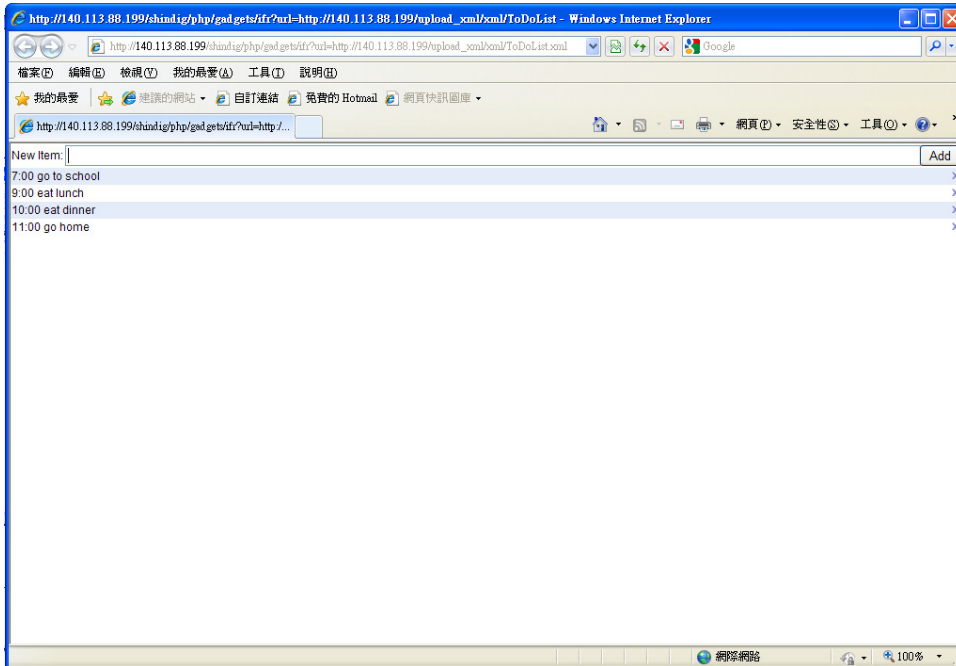


Figure 5-9 To-do-List Widget in desktop browser

Step 3: Switch to the mobile phone, it can load all items we have entered successfully; it means the function of synchronization is normal. Then, we add a new item in the top filed.

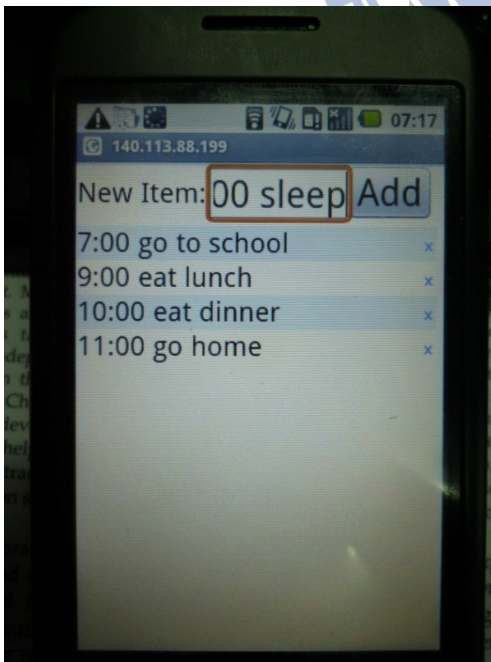


Figure 5-10 Add new items on mobile browser

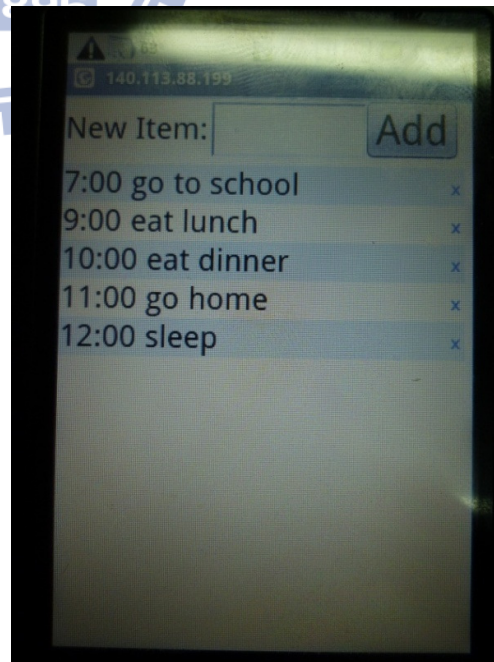


Figure 5-11 Completed the add action

Step 6: Switch to the desktop's screen, it shows the item we added
=> It means the function of real time synchronization is normal

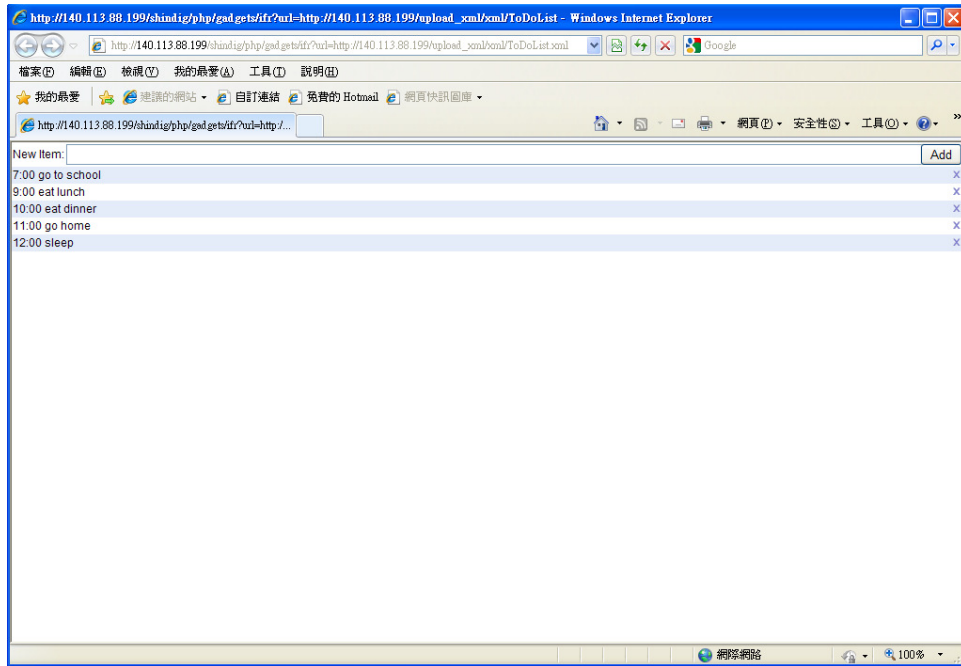


Figure 5-12 Real time synchronization

5.2.2 Test the off-line operation

Step 1: Set the phone offline → open the mobile phone's Settings tool.

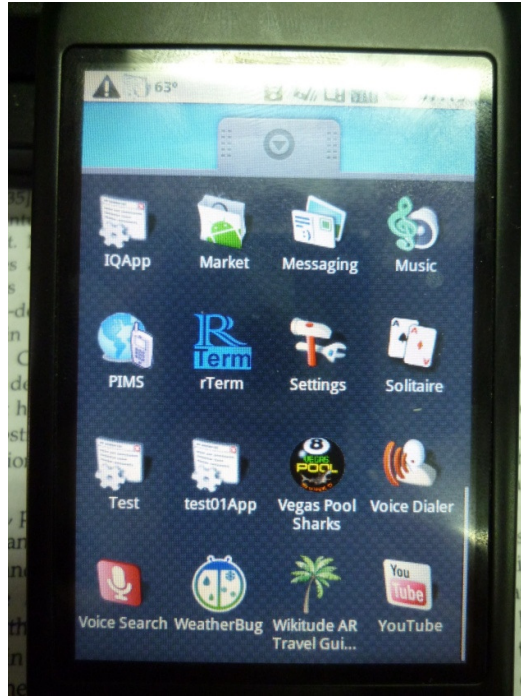


Figure 5-13 Setting toolbar of mobile phone

Step 2: cancel the green Wi-Fi check mark to abort the connection.



Figure 5-14 The network status of mobile phone



Figure 5-15 The network of mobile phone has been shut down.

Step 3: In the offline state, go into the user interface, the user has selected two widgets. Click the left icon to enter the To-Do-List page



Figure 5-16 Offline users' interface.

Step 4: In offline state, this widget will load data from local server's database. Since then, we can try to add a new item to the list.

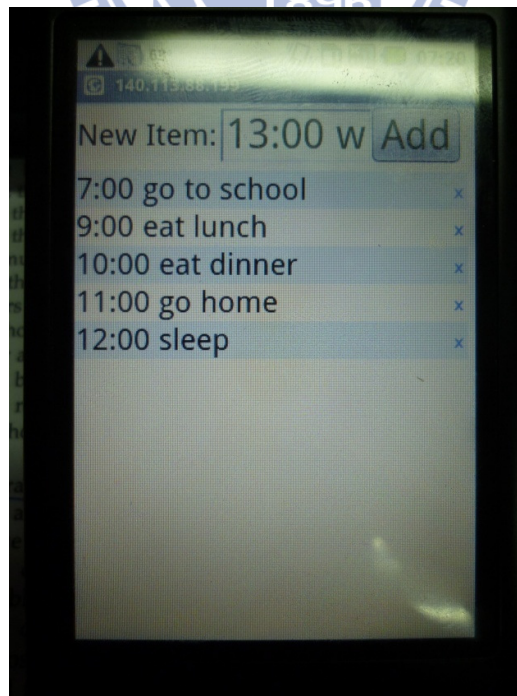


Figure 5-17 Demo of To-Do-List widget

Step 5: After we add an item to the list in offline state, it still work normally.

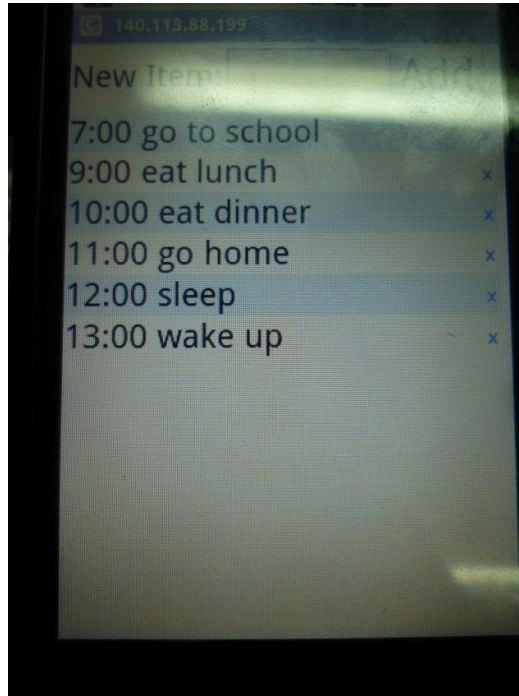


Figure 5-18 Functional testing

Step 6: re-open the connection.

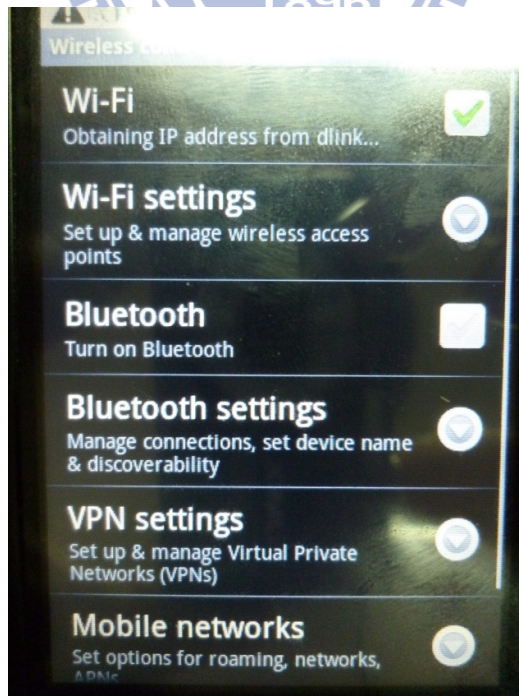


Figure 5-19 Connection Setting

Step 7:

After connecting, the mobile phone will be automatically synchronized with the server's database. It can be seen from the desktop browser, it appears the new item we added in mobile phone automatically.

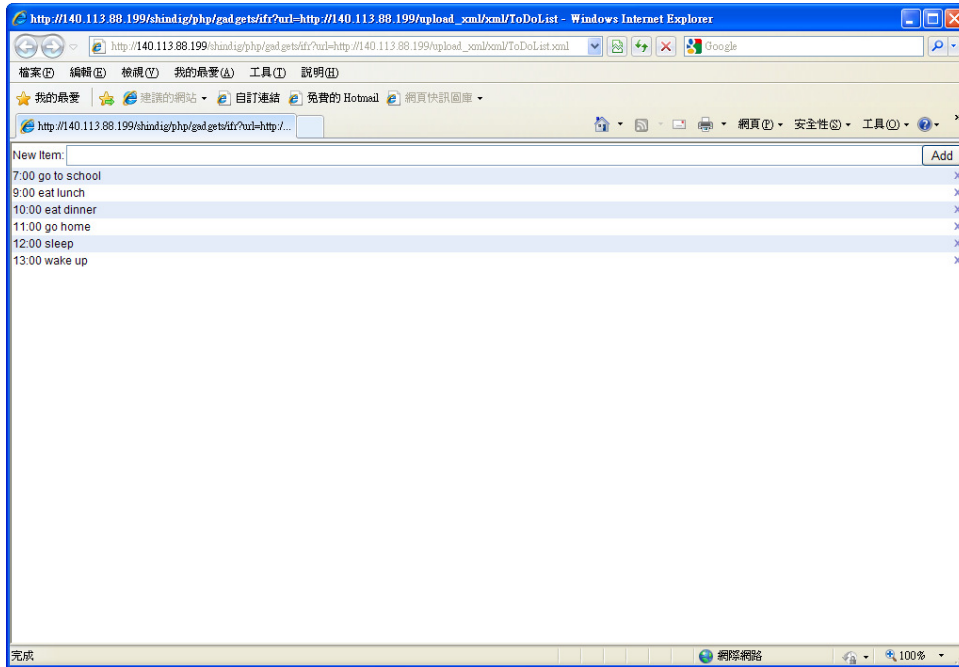


Figure 5-20 After synchronization

5.3 The Demo of RSS type Widget

Step 1: From the user's interface, click on RSS icon, it will appear Google Gears prompt message, select "I trust this site", press the lower right corner **【allow】**, so we can enable offline functionality.

When RSS Widget loading, it will appear "On-line Request" message to tell us it is online state. After caching all offline data, it will also show a message to notify us.

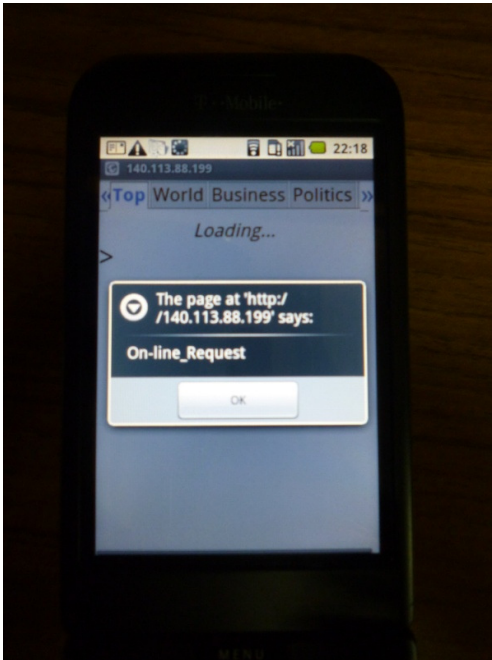


Figure 5-21 On-line Request Message

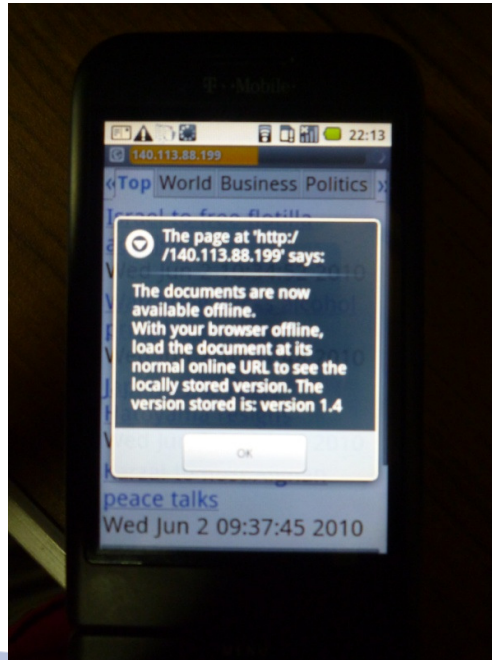


Figure 5-23 Cache finish Message

Step 2: When it is online state, go to remote webpage, it will show the real RSS content of this webpage.

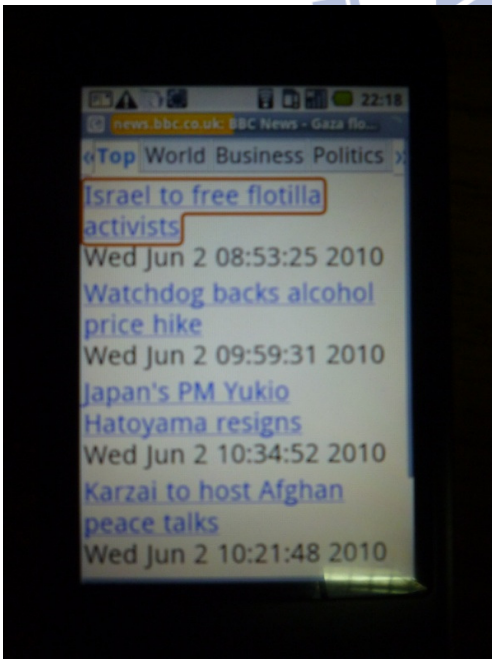


Figure 5-23 Go to the remote webpage

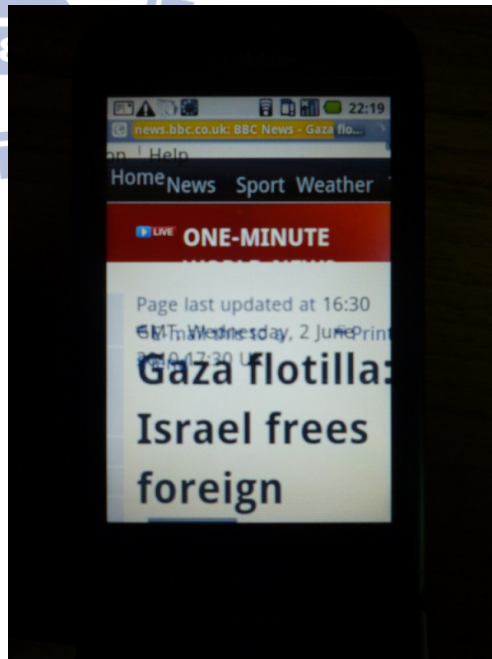


Figure 5-24 Shows the real content

Step 3: When RSS Widget loading in offline state, it will appear “Off-line Request” message. If we go to the remote webpage, it will show the tailed page now.

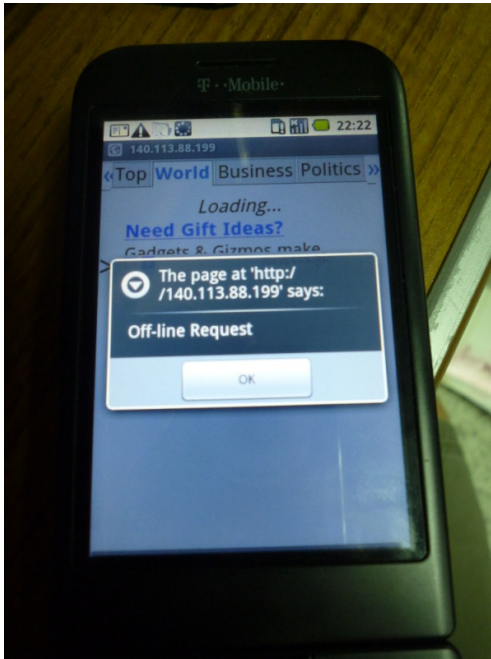


Figure 5-45 Off-line Request Message

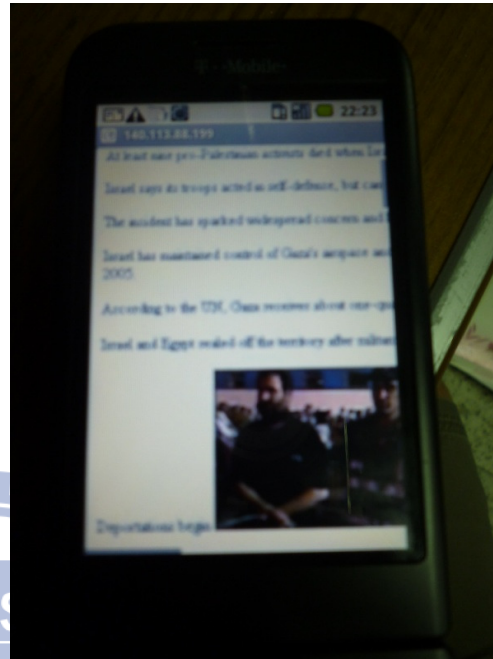


Figure 5-26 Shows the tailored content

5.4 System Evaluation

Through the previous chapters and the demo of this chapter, we should be able to clearly understand the contribution of this thesis. In addition to cross-platform and easy maintenance features, we can through the following table to compare the differences with other platforms. As we use the widget engine Apache Shindig is implemented by server-side, so the action to render Widget does not use the mobile device's resources, in other word, we can save the mobile CPU's utility. In addition, in such a situation, if the Widgets have used External API, we can use some way to make offline information be presented in the future. In addition, widget engine is

implemented by server-side can has the portable ability; we can get our information from anywhere, the server will store what widget we use and what data we store. Although, igoogole itself has many similar characteristics with us, but it has one of the most deadly shortcoming is the lack of off-line use, this shortcoming is not conducive to mobile device; that is our greatest intention quoted Google gears mechanism.

	Window mobile Widget	Yahoo! Go	Opera Widget	igoogole	Our System
Cross Platform	No	Yes	Yes	Yes	Yes
Cross Browser	No	No	No	Yes	Yes
Mobility	No	No	No	Yes	Yes
Easy to Maintain	Yes	No	Yes	Yes	Yes
Render Widget/ Save CPU utility	Local	Local	Local	Server	Server
Use External API	No	No	No	Yes	On-line: Yes Off-line: Extension
Off-line Use	Yes	Yes	Yes	No	Yes

Table 5-1 Comparison table of different platform

5.5 Usability test

When this system is completed, we have considered the works of “Jakob Nielsen” [32] and do a Usability test for users to see if our system can be accepted by general users or not. In addition, the content of these questions is referred to Benbunan-Fich’s thesis [31]. The thesis referred to what the establishment of commercial web content

should be aware. Therefore, we use these issues as our guideline. From the result of Figure 5.2, except some users were dissatisfied for the repetitive message to present on-line status, on the whole, ours system was favorably reviewed.

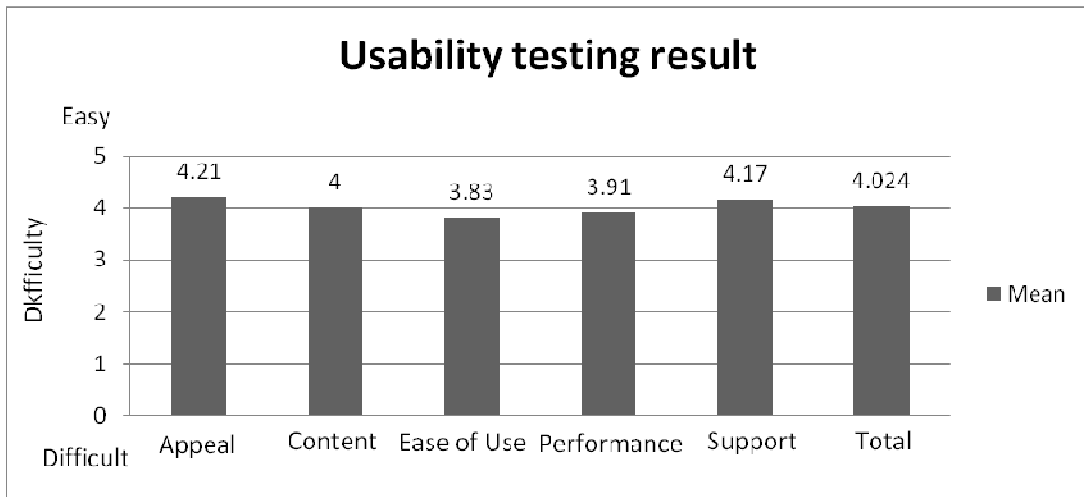


Table 5-2 Means and standard deviations from the "Usability Score"



Chapter 6

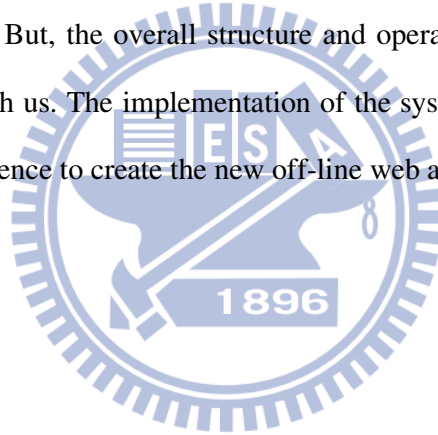
Conclusion and Future Work

In our system, we create a Widget engine environment to achieve cross-platform & cross-browser feature and let the applications allow off-line operation. And we provide Sync service to maintain mobility property. Moreover, we add Mobilize service to enforce the possibility of off-line use. But, this system is just proposed a prototype, and does not complete all of the widget functions that we can use, such as: The widget includes Google Translate API can [30] not be used in offline state, because this kind of API is another type of web service, unless we can complete the same function in client-side, or this service is difficult to use in our system. But there is a way to try in the future is that we just store the content we have been indexed (or searched), then rewrite the API. It still provides the original service in on-line state. But when offline, we can use some simple way corresponds to the on-line behavior. For example: when we use Google Translate API in off-line state, the web application will first check whether this has been the search terms, and the existence of this record stored in the local database. If do so, the web application can be just shown the result from local machine. The same way are equally available in the Google Maps API, etc.

A difficulty of development we encountered is that our system is used by JavaScript to write the program, although there is “Script Debugger for Windows” or “JavaScript debugger for Firefox” as a debug tool. However, when it executes on mobile device or desktop sometimes still be different or the debugger cannot determine the behavior of errors. So, we really need a good debug tool for mobile environment to develop the off-line web application. It will be easier for engineer to develop such applications as

we made. And some problems like browser cache, etc. We also need some good mechanism to remind the system developers the related errors may occur.

In addition to the above mentioned that we need to reinforce the integrity of the widget. Google in this year (2010) has also been announced that it won't develop Google gears in the future. The reason is the HTML5 coming of the future. The spec for HTML5 have offline storage and will be implemented the related issues including in the new generation of browser. However, as HTML5 is not universal, and there is no mobile browser that supports. Therefore, the implementation of our system still uses Google Gears to finish off-line operation. Although the future HTML5 will be replaced Google Gears. But, the overall structure and operation of the system won't be compared too far with us. The implementation of the system and the problems we faced also can be a reference to create the new off-line web application.



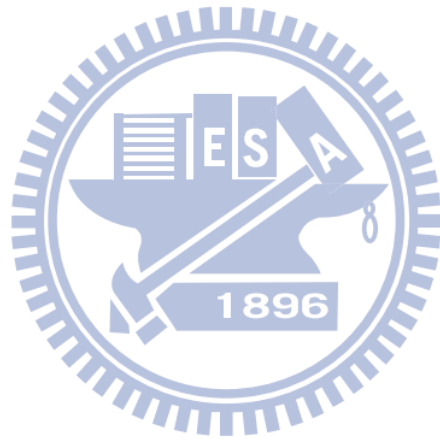
Reference and Bibliography

- [1] Web applications, Retrieved July 16, 2010, from http://www.w3.org/2008/webapps/wiki/Main_Page
- [2] Java ME, Retrieved July 16, 2010, from <http://java.sun.com/javame/technology/index.jsp>
- [3] Windows Mobile, Retrieved July 16, 2010, from <http://www.microsoft.com/windowsmobile/en-us/help/default.mspx>
- [4] Symbian, Retrieved July 16, 2010, from <http://developer.symbian.org/>
- [5] BREW, Retrieved July 16, 2010, from <http://brew.qualcomm.com/brew/en/developer/overview.html>
- [6] Mobile Linux, Retrieved July 16, 2010, from http://www.linuxfoundation.org/collaborate/workgroups/mobile-linux/mobile_platform_guidelines_summary
- [7] Android, Retrieved July 16, 2010, from <http://code.google.com/intl/zh-TW/events/io/2010/sessions/beginners-guide-android.html>
- [8] Android market, Retrieved July 16, 2010, from <http://www.android.com/market/#app=com.epocrates>
- [9] Iphone App, Retrieved July 16, 2010, from <http://developer.apple.com/programs/iphone/develop.html>
- [10] Windows Mobile Widgets, Retrieved July 16, 2010, from <http://msdn.microsoft.com/en-us/library/dd721906.aspx>
- [11] Yahoo! GO, Retrieved July 16, 2010, from <http://mobile.yahoo.com/go>
- [12] JavaScript & AJAX, Retrieved July 16, 2010, from

<http://www.w3.org/standards/webdesign/script.html>

- [13] XML, Retrieved July 16, 2010, from <http://www.w3.org/TR/REC-xml/>
- [14] Widgets, Retrieved July 16, 2010, from <http://www.w3.org/TR/widgets/>
- [15] Google Gears, Retrieved July 16, 2010, from <http://gears.google.com/>
- [16] RSS, Retrieved July 16, 2010, from <http://cyber.law.harvard.edu/rss/rss.html>
- [17] Yahoo Widget, Retrieved July 16, 2010, from <http://widgets.yahoo.com/>
- [18] Dynamic Web Content, "The Information Revolution", J. R. Okin. ISBN 0976385740. Ed. Ironbound Press, 2005. 350 pp
- [19] Same Origin Policy, Retrieved July 16, 2010, from <http://taossa.com/index.php/2007/02/08/same-origin-policy/>
- [20] Html 5, Retrieved July 16, 2010, from <http://www.w3.org/TR/2010/WD-html5-20100304/>
- [21] Chi-Yang Tsai, Shyan-Ming Yuan, "Web Page Tailoring Tool for Mobile Devices", 國立交通大學，電資學院碩士班論文，民國 95 年 6 月
- [22] Dom tree, Retrieved July 16, 2010, from <http://www.w3.org/DOM/>
- [23] Muffin server, Retrieved July 16, 2010, from Retrieved July 16, 2010, from <http://muffin.doit.org/>
- [24] Silverlight, Retrieved July 16, 2010, from <http://www.silverlight.net/>
- [25] Sun JavaFX, Retrieved July 16, 2010, from <http://java.sun.com/javafx/>
- [26] SiteShoter, Retrieved July 16, 2010, from http://www.nirsoft.net/utills/web_site_screenshot.html
- [27] D. Crockford. The application/json media type for javascript object notation (JSON). Request for Comments 4627, The Internet Society, July 2006.
- [28] Firebug, Retrieved July 16, 2010, from <http://getfirebug.com/whatisfirebug>
- [29] Tung-Hing Chow, Shyan-Ming Yuan, "An Offline Browsing Mechanism for Mobile Devices", 國立交通大學，電資學院碩士班論文，民國 98 年 6 月

- [30] Google Translate API, Retrieved July 16, 2010, from
<http://code.google.com/intl/zh-TW/apis/ajaxlanguage/documentation/>
- [31] Benbunan-Fich, R. "Using protocol analysis to evaluate the usability of a commercial Web site." *Information & Management* (39) 2001, pp 151-163.
- [32] Jakob Nielsen, *Usability Engineering*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1995



Appendix A:

System Usability Scale

	Strongly disagree					Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
2. I think this user interface is suitable for small screen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
6. I think it's easy to find the button I wanted to press	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
7. I can figure out path I've completed	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
8. I think the system has a good performance	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
10. I think this system give me more convenience	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	