Developing Offline Web Application

Kanda Runapongsa Saikaew[1], Art Nanakorn[2], Thana Pitisuwannarat[3]
[1,3]Department of Computer Engineering [2]Microsoft Thailand
Khon Kaen University 87/2 Wireless Road, Phatumwan
Khon Kaen, Thailand, 40002 Bangkok, Thailand, 10330
krunapon@kku.ac.th[1], i-arnana@microsoft.com[2],t.pitisuwannarat@gmail.com[3]

Abstract

Nowadays a large number of software is increasingly available in the form of web application. The web is where most software is moving for cost, convenience, agility, and increased overall business value. Millions users rely on web application to perform their work such as using web mail and web calendar. On the other hand, Internet is not always reliable. Therefore, web application users are seriously affected by disconnected Internet connection. Developing web application that can be accessed both online and offline is thus necessary. In this paper, we propose the implementation of typical web application that still provides users service despite of being disconnected. This web development using only open source tools which include Gears, Javascript, MySQL, and PHP.

Key Words: Offline web application, Gears, Open source

1. Introduction

In modern times, human daily activities almost cannot live without web application in many ways. They need to access web application to check their e-mails, browse their calendar appointments, prepare presentations with their online tools, update with the latest news, or chat with their friends.

There are many reasons why web application has been extensively widespread. People do not need to have their own computers to use web application. Students can use computers at school labs, office workers can use ones at their offices, and other groups of people can use ones at Internet Café. Not only web application is suitable for people who do not have their own personal computer, it is also appropriate for people who have multiple machines. Furthermore, people can browse their web application through mobile devices, such as a smart phone and a PDA.

Web application can be viewed as its own computing platform. It is accessible no matter what operating systems of the machines are. It is coded in browser executable language such as HTML, PHP, Java, and JavaScript. Using web browser to access the application provides cross-platform compatibility.

From the view of software developers, maintaining web application is much easier than maintaining desktop application. The developers do not need to announce the release of the next version, make the CD/DVD for software installation, and most importantly test the upgraded software on many platforms that clients may use.

The number of web application users can be estimated by the number of Internet users. According to the report of Internet Usage Statistics [1], as of December 31, 2008, the total number of Internet users is 1,574,313,184 with the growth rate as 336.1 % during the years 2000-2008. As the statistics show, the number of Internet users is extremely large and growing fast. Any company that can sell their products and services online will easily make huge profit. The example of such company is Google Inc. which still has its revenue grew 18 percent, to $5.7 billion for the fourth quarter of 2008 despite that this period is during the deepening global recession [2].

When using traditional web application, users have to go online. Consider a business man who uses online applications to access his emails, keep a tab of his appointments and to store his contact information while he travels to meet with his customers. Since he will be travelling most of the time, he may not access to Internet connection all the time. When he is offline, he can no longer access his favorite online applications.

In real situation, it is possible that we cannot access the web application because of several reasons such as overloading web server and unreliable

Internet service provider. Therefore, accessing web application when users are offline becomes important.

W3C also realizes the need for offline web application as it plans to include the support of offline web application in HTML5 [3]. The specification aims to address this by providing two solutions: 1) a SQL-based database API for storing data locally and 2) an offline application HTTP cache to make sure applications are available even when the user is disconnected to their network. Currently Firefox 3 [4] is planning to support these capabilities by implementing online/offline events. It will be interesting to see how the other browsers support this specification.

Several tools are offered in developing offline web application, such as using Gears [5], Microsoft Silverlight [6], Adobe Air [7].

Gears, developed by Google, is an open source browser extension that enable offline web applications. Developer can call Gears APIs to create an offline web application by using Javascript. While Gears is free and open, other tools are proprietary.

Microsoft Silverlight plans to include offline feature in the future. Developers also have to know .NET programming technique instead of typical web application development like Gears.

Adobe AIR, based on Flash technology, enables application to be able to work offline but developers must have specific knowledge about Flash and AIR platform.

In term of developers support and community, Gears has provided several updated tutorial and samples. It also has active and well-participated Gears discussion groups as well as Gears blog to keep up to date on Gears.

Therefore, we choose to develop offline web applications using Gears since it is an open source, consists of complete features to enable offline application, and has great online resources for developers.

In this paper, we present our developed offline user registration web application using Gears which can operate when users are in both online and offline modes. We choose to implementation registration web application because it is common and widely used in many real world applications.

Although there have been several web applications that make offline using Gears, we have not found sample codes of a simple web application work offline with the codes that synchronize local data with server data. In Gears API Documentation and manual (http://code.google.com/apis/gears/), such sample also does not exist. Thus the explanation of the development should be beneficial to several web developers. Furthermore, the described implementation approach can apply to other types of web application easily. Figure 1 illustrates the relationship between the Gears-enabled client and the web server

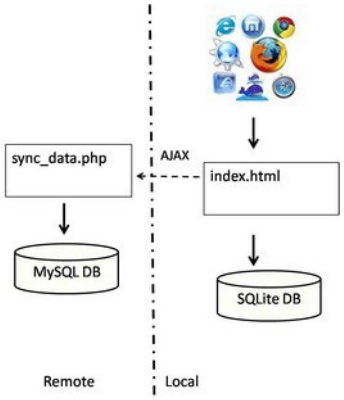

Figure 1. Gears-enabled client and server

The structure of the paper is as follows. Section 2 describes about background and related work. Section 3 then explains about web application development followed by experimental result in Section 4. Finally, we conclude in section 5.

2. Background and related work

Gears is an open source web browser extension that enables users to use web application even though they are in offline mode and enables more powerful web applications features. Nevertheless, the goal of Gears is not just to enable offline application, but to bridge the gap between web application and desktop application. Users can view the browser as a standard, but powerful, virtual machine for applications which make the operating system irrelevant.

To use Gears [8], users only need to install Gears browser extension. Using Gears, a web application can cache the data when the web application is offline. It also can synchronize the data in cache with the data at the server when it is online again. It also consists of a local server which is to cache and provide resources without needing to contact a server.

Specifically, Gears modules [9] include 1) Database module, which can store data locally by using SQLite. 2) WorkerPool module, which provides parallel execution of JavaScript code 3) LocalServer module, which caches and serves application resources such as HTML, JavaScript, and images 4) Desktop module, which lets web

applications interact more naturally with the desktop and 5) Geolocation module, which lets web applications detect the geographical location of their users.

Gears has enabled many real web applications to work offline such as YouTube, WordPress, Google Reader, and Gmail. In this section, we will describe some of such applications as follows:

YouTube uses Gears as an alternative for users to upload their large size videos [10] as shown in Figure 2. Gears breaks typical HTTP file upload limitations such as limited file size, time and so on. Using Gears, users can now upload files larger than 100 MB by using just their Gears enabled web browsers.

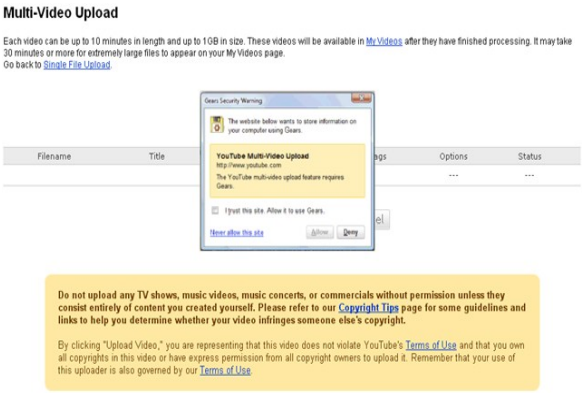

Figure 2. YouTube video upload with Gears [10]

With Gears, WordPress can speed up every page load and enable users to blog faster. WordPress with Gears will store images, JavaScript libraries and CSS files of admin page on the client local hard drive and will not download them from the web server each time client requests. Thus, it should speed up when the connection traffic is busy.

Google itself also exploits Gears in their products, such as Google Reader, Google Docs, and Gmail. With Gears, Google Reader allows subscribers to access the RSS feeds even when they are offline. Using this technology, Google Reader enables users to store 2,000 RSS feeds on their local machine to read it any time [12]. They can also tag or star the feeds and which will be synchronize back to the server automatically when the client goes online.

Recently, Google rolls out a Gears version of Gmail that will be available to users. Using browser with Gears plug-in, Gmail detects when users are offline. It caches users' e-mails so that users can read, reply, search, add stars, and label them. When users are back to the Internet again, Gmail sends all the messages. Users can also open attachments. [13]

As we see from many sample web applications previously described, Gears enable web applications to work offline and faster. Although Gears are useful for almost all web applications, it is unsuitable for real-time web applications that their data is continuously updated such as the web application that monitors and displays traffic.

3. Web application development

In this section, we will describe about the development of our web application which is to accept registration from students for attending a training. We choose to illustrate this web application because registration process is generally in almost all web applications. In the following subsections, first we present the web interface and its architecture then we explain server-side script and finally we present client-side script. The interface of this web application is shown in Figure 3.

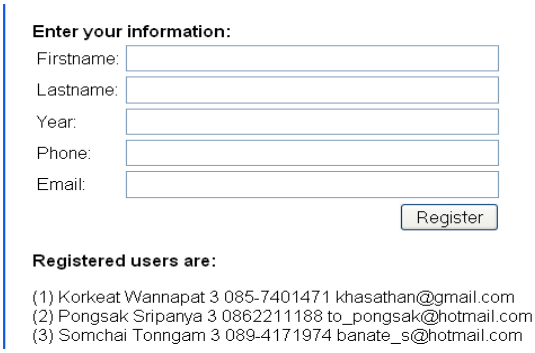

Figure 3. Registration web application

From Figure 3, this web application will ask the user to fill out the information which includes first name, last name, year, phone, and email. After the user fills out all information, and press button "Register", the list of registered users will be updated as shown on the bottom part of Figure 3. The user information will be saved in a local database. When the user browser is connected to the Internet, the application will be synchronized with other users' information in the server database.

To develop an offline web application using Gears, we need to install Gears plug-in with his browser. We also need to check whether the client's browser has been installed with Gears plug-in. Then,we need to call Gears APIs from his client-side script. The client-side script has to call Gears Database API in order to create a local database. To call server-side script asynchronously, we use Gears HttpRequest API. In addition, the client-side script needs to include the codes that copy the data from a local database to a server database and vice versa in

order to synchronize data. We copy local data to a server database whenever the new data has been inserted and copy server data to a local database at the beginning of the client-script code.

The development details are as follows:

1. Install Gears plug-in [14]. We need to detect whether or not Gears is installed on a user's system before calling the APIs and also to determine when to display an installation prompt to the user. A web application developer using Gears should always initialize Gears using gears_init.js which can be downloaded [14].

```
<script src="gears_init.js"></script>
<script>
  if (!window.google || !google.gears) {
    location.href= "http://gears.google.com/?
action=install&
message=<your welcome message>" +
                "&return=<your website url>";
  }
</script>
```

Figure 4. Code to check Gears installation

2. Store data in a local database. In this step, we use Gears Database API to store data in SQLite database system. Figure 5 shows the code that we use to open a database and create a table.

```
try {
     db =
google.gears.factory.create('beta.database');

     if (db) {
        db.open('db');
        db.execute('create table if not exists users' +
                ' (firstname varchar(255),
             lastname varchar(255), year integer,
             phone varchar(255),
             email varchar(255), timestamp int)');
        ….
} catch (ex) {
   alert(ex.message);
}
```

Figure 5. Code to use a local database

3. Develop the server-side script. The script is to access and update data at a server database. It accepts a parameter called "method" and then call SQL statements according to the method value.

```
<?php
   ..
   // connect database system and select database
   mysql_select_db("registerdb")
   or die(mysql_error());
   if ($method == "access") {
        // use SQL statements to retrieve data
   } else if ($method == "update") {
        // use SQL statements to insert data
   }
?>
```

Figure 6. Server-side script to manage data

4. Synchronize data between server and local databases. In synchronization, we copy data from a server database to a local database and copy newly inserted data from clients to a server database.

4.1 Copy data from the server database to the local database. In this process, there are two main steps. The first step is to call server script to retrieve data from the server database. In this program, we use HttpRequest API [14] to call the server side script with method="access" which means that we want to access or read server data. The second step is to tokenize data received from the server database and insert such data into the local database. We use split PHP function to separate different records in the same response and to separate columns in the same record. We write the code to process data received from server which is when the readyState of Gears HttpRequest is equal to 4.

Figure 7 shows partial code that uses HttpRequest API to access sthe erver-side script and insert data from the server database to the local database.

```
var sync_script =
'http://host/directory/sync_data.php';
…
var request =
google.gears.factory.create('beta.httprequest');
request.open('POST', sync_script);
request.setRequestHeader("Content-
type","application/x-www-form-urlencoded");
request.onreadystatechange = function() {
if (request.readyState == 4) {
   var data = request.responseText;
   var rows = data.split("<br/>");
   // process each row data from server script
   // check if data exists in local database
   // if it doesn't exist, insert data received
   db.execute('insert into users(firstname, lastname,
year, phone, email, timestamp) values
(?, ?, ?, ?,?,?)', [f,l,y,p,e,t]);
   …
```

```
};
request.send(params);
```

Figure 7. Code to copy data from server to client

4.2 Copy data from local database to server database. In this process, there are two main steps. The first step is to form parameters of server script to insert data in server database. The second step is to call server script with method = "update" which means that the client code would like to update data by inserting new rows in the local database.

```
var request =
google.gears.factory.create('beta.httprequest');
var
params="method=update&f="+firstnames[index];
params += "&l="+lastnames[index];
params += "&y="+years[index];
params += "&p="+phones[index];
params += "&e="+emails[index];
…
request.open('POST', sync_script);
request.setRequestHeader("Content-
type","application/x-www-form-urlencoded");
request.send(params);
```

Figure 8. Code to copy data from client to server

4. Experimental Result

In the experimental result, we need to illustrate that each client will see the same application data despite that he/she also stores data locally on his/her computers. In effect, this tests whether we can synchronize data which happens when users are online. We also need to show that the web application can work offline. Thus, the experimental result is divided into two scenarios: 1) Testing whether the application can store data locally and synchronize with data at server when it is online and 2) Testing whether the application can work offline.

4.1 Synchronizing Data When Online

We have set up scenario that two concurrent users being registering on the web application. Figure 9 shows user “Art” fills the registration form while there are three registered users. In the meantime, user “Thana” is also trying to register when there are also three registered users as shown in Figure 10.

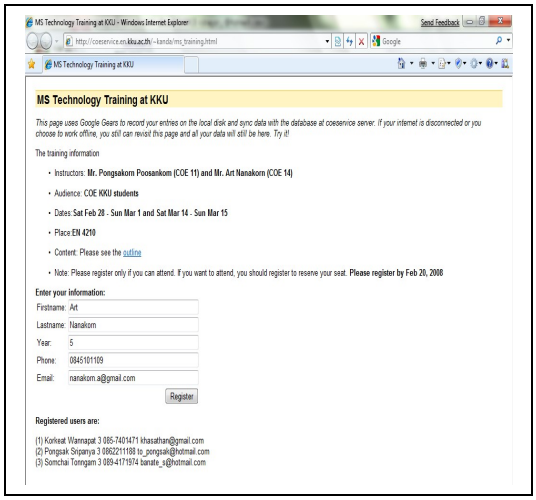

Figure 9. User “Art” is regisering

After both users made registration, user “Art” refreshes the page and found records of his newly registered users and the total number of registered users is six as shown in Figure 11. User “Thana” also found the same result that user “Art” found. This is shown in Figure 12.

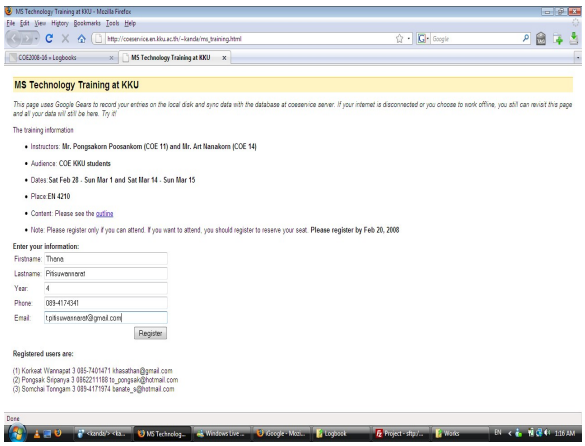

Figure 10. User “Thana” is also registering

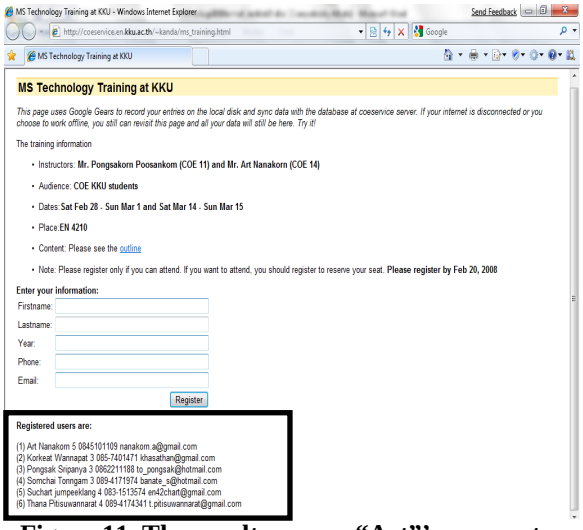

Figure 11. The result on user "Art"'s computer

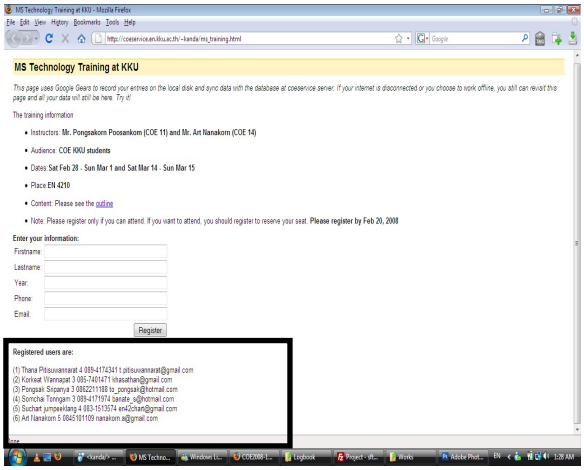

Figure 12. The result on user "Thana"'s computer

4.2 Using the Application When Offline

Figure 13 shows that a client can still see the web application with his browser while it is in work offline mode. Figure 14 illustrated that the user still can access and fill the web application form.

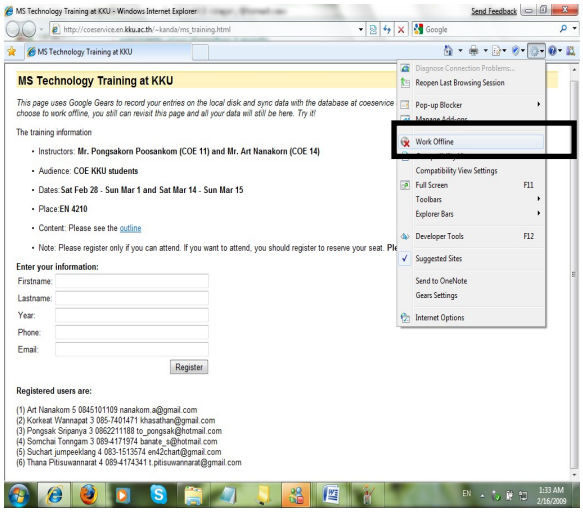

Figure 13. Browser with Work Offline mode

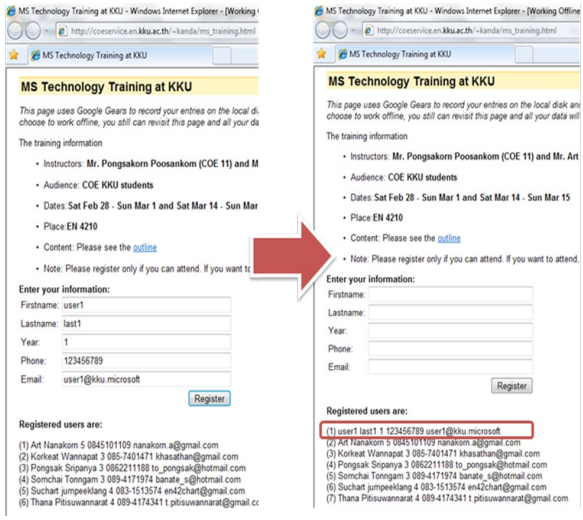

Figure 14. Using web application while offline

5. Conclusion

In this paper, we have presented an approach to develop offline web application that can be applied to typical web application. Web application is significantly used widely by thousand millions of users while the network connectivity sometimes unreliability and slow. Thus, web developers should have the ability of enable web application to work offline and improve performance by using existing open source tools, such as Gears.

There have been many web applications that work while offline but these web applications are developed by a large software company or organization such as Google and WordPress. In this paper, we illustrate techniques in making typical web application offline with open source tools. Our approach can be applied to many web applications

that are developed by a single developer or a small group of software house. In the future, we are interested to improve the performance of web application by using WorkerPool module.

References

[1] Miniwatts Marketing Group. "World Internet Usage Statistics News and World Population Stats" [online] 2008 [cited 2009 Feb 15]. Available from: http://www.Internetworldstats.com/stats.htm

[2] Miguel Helft, "Google Beats Forecast Even as Its Profit Tapers" [online] 2009 [cited 2009 Feb 15]. Available from: http://www.Internetworldstats.com/stats.htm

[3] W3C Working Group, "Offline web Applications" [online] 2008 [cited 2009 Feb 15]. Available from http://www.w3.org/TR/offline-webapps/

[4] Nickolay Ponomarev, "Online and offline events" [online] 2008 [cited 2009 Feb 15]. Available from https://developer.mozilla.org/En/Online_and_offline_events

[5] Google, "Gears: Improving your browser" [online] 2008 [cited 2009 Feb 15]. Available from http://gears.google.com/

[6] Microsoft, "The Official Microsoft Silverlight Site", [online] 2008 [cited 2009 Feb 15]. Available from http://silverlight.net/

[7] Adobe, "Adobe AIR" [online] 2009 [cited 2009 Feb 15]. Available from http://www.adobe.com/products/air/

[8] Sriram Balaji, "Infosys | Microsoft: Offline web Applications" [online] 2008 [cited 2009 Feb 15]. Available from http://www.infosysblogs.com/microsoft/2008/08/offline_web_applications_1.html

[9] Wikipedia, "Gears (Software)" [online] 2009 [cited 2009 Apr 10]. Available from http://en.wikipedia.org/wiki/Google_Gears

[10] Shoaib Hashmi, "'Google Gears' now lets you upload YouTube videos up to 1 GB" [online] 2008 [cited 2009 Feb 15]. Available from http://startupmeme.com/google-gears-lets-you-upload-youtube-videos-upto-1-gb/

[11] James, "Wordpress 2.6 and Google Gears:geniosity musing" [online] 2008 [cited 2009 Feb 15] . Available from http://www.geniosity.co.za/musings/wordpress/wordpress-26-and-google-gears/

[12] Google, "Google Reader – Offline reading" [online] 2009 [cited 2009 Feb 15]. Available from http://www.google.com/help/reader/offline.html

[13] Erick Schonfeld, "Gmail Goes Offline with Google Gears" [online] 2009 [cited 2009 Feb 15]. Available from http://www.techcrunch.com/2009/01/27/gmail-goes-offline-with-google-gears/

[14] Google, "Gears API – Google Code" [online] 2009 [cited 2009 Feb 15]. Available from http://code.google.com/apis/gears/architecture.html