CENTER FOR CONVERGENCE AND EMERGING NETWORK TECHNOLOGIES – CCENT

Syracuse University

TECHNICAL REPORT: T.R. 2014-003

Accessing External Databases from Mobile Applications

Version 2.0

Authored by: Anirudh Nagesh, Keshav Khandelwal , Carlos E. Caicedo

June / 2014

Document history:

Version 2.0: Update to v1.0 document by Keshav Khandelwal, edited by Carlos Caicedo

Report submitted on: 6/22/2014

Publicly available on: 6/23/2014

Version 1.0: Written by Anirudh Nagesh, edited by Carlos Caicedo

Report submitted on: 12/09/2011

Publicly available on: 3/21/2012

Abstract:

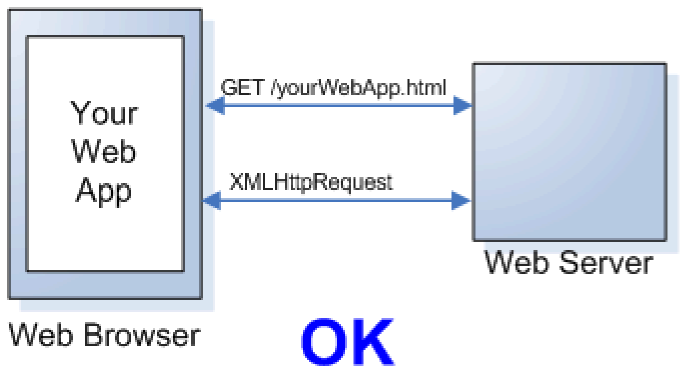
In the early years of this decade, we have seen an explosion in the number of mobile devices such as smart phones and tablets from varied manufacturers consisting of varied operating systems. At the same time, the number of mobile applications has exceeded 500,000 across all platforms. One of the characteristics of several of these applications is their use of databases (either local or remote) for accessing data. Accessing data from remote databases in mobile applications is not straightforward. Database queries cannot be invoked on a remote database as done on a local database. In this paper, we will discuss the approach to solve this problem by using MVC (Model-View-Controller) software design pattern. By using this approach, we can enable mobile applications to communicate with remote databases seamlessly. The solution discussed in this paper is platform agnostic i.e, this solution can be used independent of the platform that is used to develop mobile applications. It can be a native app(iOS, Android), a mobile web app(HTML5, jQuery, jQtouch) or an Hybrid app(Phone gap, Rho Mobile, Titanium).

1 Introduction:

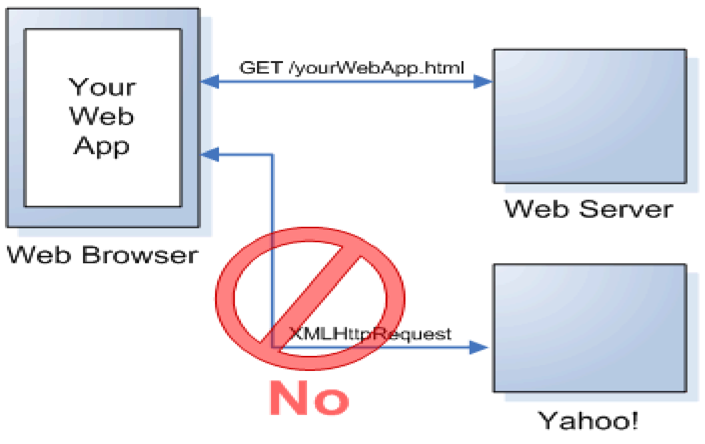
Some of the popular operating systems which power mobile devices are Android, iOS, Windows 7, Symbian, Palm OS etc. Adding to these list also include some of the cross platform libraries such as jQuery, jQtouch, Sencha, Titanium, HTML5 etc. This has resulted in an exponential growth in the number of mobile applications available across all platforms. The applications can be categorized into Games, Enterprise Apps, Educational apps, mobile web apps, information based apps etc. Most of these applications have lot of data either included in the app itself or hosted in a remote database which assists in the provision of the services and/or capabilities that the application is designed for. However, accessing data in a database from a mobile application is not straight forward.

2 Architectural Design:

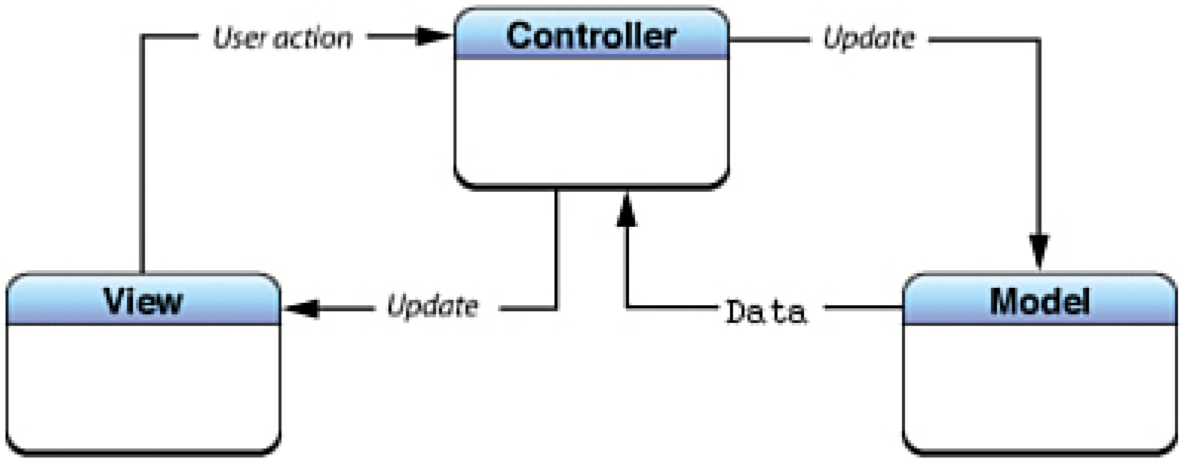
Accessing data on web server’s database can be accomplished by using XMLHttpRequests or XHR requests, but there are limitations imposed by cross-domain scripting. The client applications which requests data from web server using XMLhttprequests can easily retrieve data if the web server is on the same domain.



However, browsers impose a restriction for any requests from the client application to a server in a different domain – for example Yahoo web services.



Hence we would need a three tier approach or a proxy) to solve the cross domain scripting problem applying the Model-View-Controller paradigm (MVC paradigm). This paradigm is a design methodology in which the mobile UI acts as View, the backend database acts as Model and the intermediate php, asp scripts act as Controller. A diagrammatic representation is shown below.

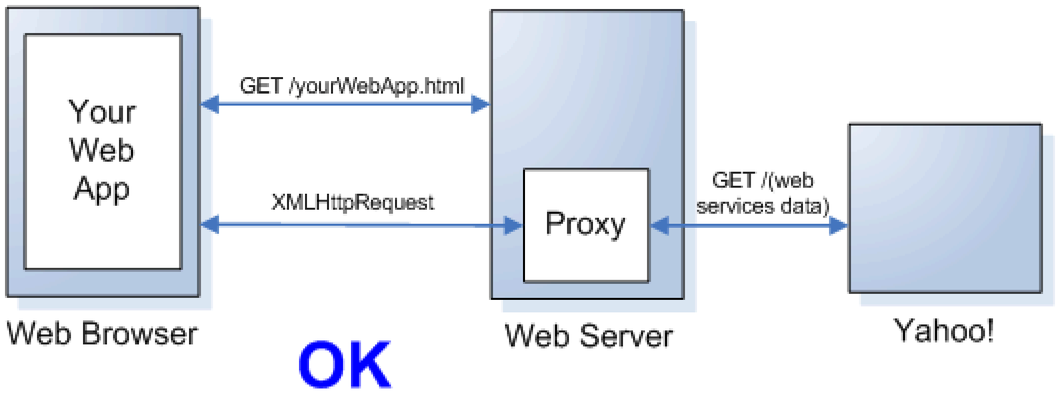


View: The front end graphical interface of the mobile application (Screens, buttons, views, table views)

Model: The back end database hosting the data (MySql, MSSql database). A typical model provides data and methods that provide information to the application. It is completely independent of how the application looks like.

Controller: A php or ashx (asp handler page) acting as an intermediate process which communicates back and forth with the database and updates information to the view. It is responsible for receiving user's input and acting accordingly. It acts like a bridge between View and Model.

By using the above approach, we can install a proxy on the web server such as a php script. Instead of trying to get data using XMLhttp requests, we pass on the request to the proxy which in turn passes on the request to the web server and retrieves the data and passes it back to the client application.



3 Experiments, findings and Analysis:

During the course of implementing and testing this MVC approach, we used the following tools to test the procedure.

1. MySQL Server: to host databases and data.
2. MSSql Server: to host MSSql databases and host data.
3. PHP (version 3.5): PHP compiler used to write php scripts acting as a proxy.
4. HTML and JavaScript (jQeury): to create the front end of the client mobile application.

MySQL Server: MySQL Server is an open source relational database management system that runs as a server providing multi-user access to multiple databases. It is primarily an RDBMS without a GUI interface. But in this research, we have used MySQL workbench which enables users to graphically administer MySQL databases and also design the same. In this research, we have used MySQL server hosted at 'ist-s-students.syr.edu', created tables and inserted data into the same. Configuration of MySQL and queries to create the tables and inserting data will be explained in the next section.

MSSQL Server: An alternative to MySQL server is MSSQL server. It is a relational database server developed by Microsoft. It provides the same functionality as MySQL server with a GUI to create database schemas, tables and execute queries. But unlike MySQL, it is not open sourced and has to be bought with a license fee. The syntax of the queries are a bit different from that of MySQL queries. Configuration of MSSQL and queries to create tables and inserting data will be explained in the next section.

PHP: PHP is a general purpose server side scripting language designed for web development to develop dynamic web pages. It can be embedded into HTML pages and can be interpreted by the web server which has PHP processor module installed. It is a platform independent language which runs on multiple platforms such as Windows, MAC and Linux. One of the strongest and most significant uses of PHP is its support of multiple databases. It can support MYSQL, MSSQL databases via an abstraction layer like ODBC (or JDBC) – Open Database Connectivity. The other advantage of using PHP as a proxy for mobile applications is its ability to output data in the form of XML and JSON. The php file can connect to the remote or local database, run queries, retrieve data and convert it into JSON or XML that acts as input to the client side mobile application. PHP also has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows) and countless others. Configuration of PHP and the sample code for the proxy file will be explained in the next section.

4 Configuration of PHP, MYSQL and MSSQL:

4.1 Configuration of PHP:

1. Download the PHP module or processor from http://windows.php.net/download/
2. Install the php by clicking the .msi file.
3. Select all the default options and finish the installation.
4. Select 'Do not set up a web server' during the installation as we will be using Microsoft in built IIS as a web server.

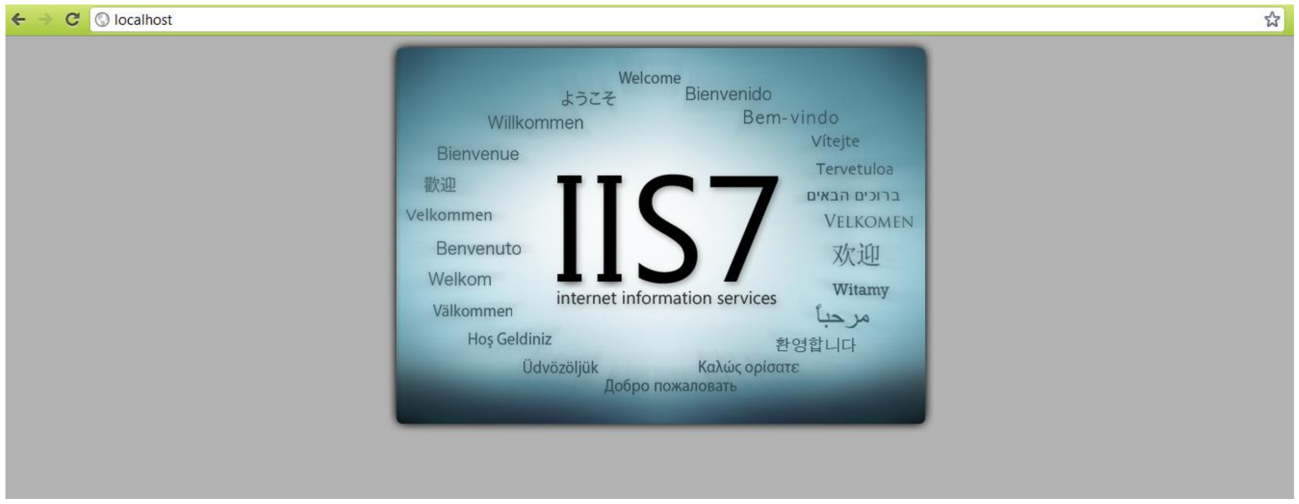
5. Go to the directory where PHP was installed -> Open the php.ini file in notepad and configure CGI – and Fast CGI- specific settings as below:

   fastcgi.impersonate = 1
   fastcgi.logging = 0
   cgi.fix_pathinfo = 1
   cgi.force_redirect = 0

6. To verify the proper installation of PHP and also the web server follow the below steps.
7. Go to Control Panel->Administrative Tools-> Internet Information Services Manager(IIS Manager)
   a. If Internet Information Service Manger (IIS) is not available inside Administrative Tools, Go back to Control Panel -> Programs and Features -> Turn Windows Features On or OFF, select Internet Information Services
8. In the "Feature View" page open the "Handler Mappings" feature -> In the "Action" pane click "Add Module Mapping…" and add the below information in "Add Module Mapping" dialog box:

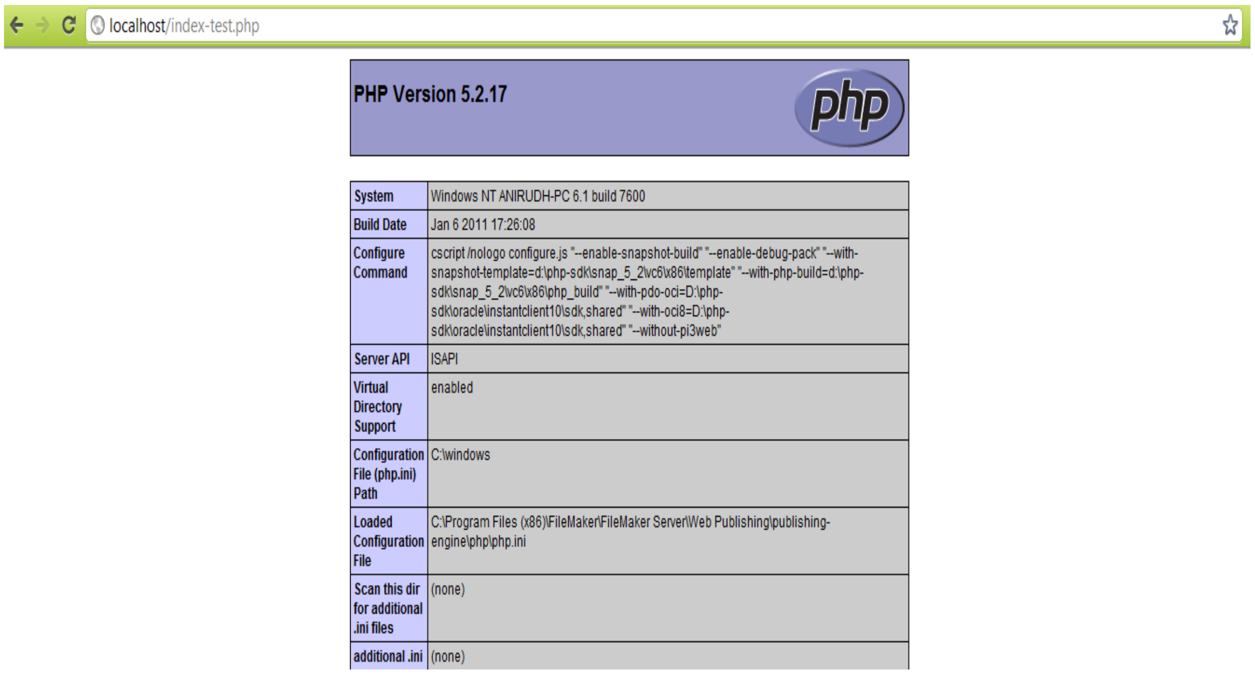
   Request path = *.php
   Module: FastCgiModule
   Executable: [DirectoryName]\[Path to PHP installation]\php-cgi.exe
   Name: PHP_via_FastCGI

9. Click on "Request Restriction" button and check "Invoke handler only if request is mapped to:" -> select File or Folder -> Click OK
10. On the right side of the window, make sure the service is running (Start command should be disabled).
11. Stop the server by clicking Stop and Restart it.
12. In the connections window, expand the tree by clicking on the root (Username –PC) and select Sites under it.
13. Click on default website option.
14. In the Actions window, make sure that the service is started. Restart it.
15. Open any Browser (For example, Chrome) and type in **http://localhost/** in the address bar. IIS information should be displayed as shown below.



16. This confirms that the IIS server is configured and running properly.

17. To test the configuration of PHP, go to C:\Inetput\wwwroot folder, create a new file and name it as index-test.php.
18. Open the index.php file in any text editor and include the below code snippet.
    <?php
    Phpinfo();
    ?>
19. Type in '**http://localhost/index-test.php'** in the browser and configuration of the php should be seen in the browser as shown below.



PHP Version 5.2.17

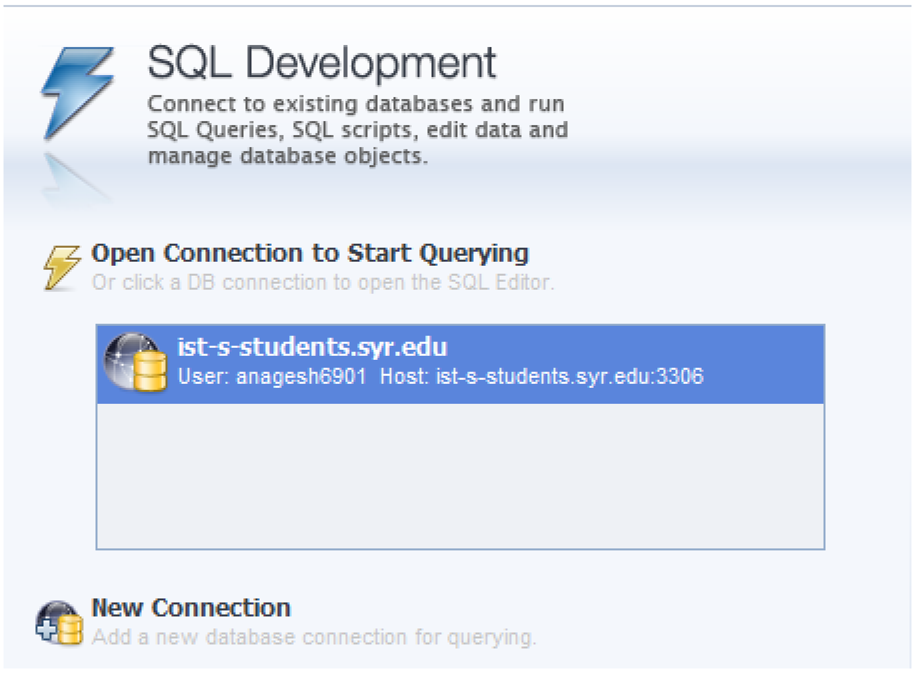
System	Windows NT ANIRUDH-PC 6.1 build 7600
Build Date	Jan 6 2011 17:26:08
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	ISAPI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\windows
Loaded Configuration File	C:\Program Files (x86)\FileMaker\FileMaker Server\Web Publishing\publishing-engine\php\php.ini
Scan this dir for additional .ini files	(none)
additional .ini	(none)

20. This confirms the proper installation and configuration of PHP on the machine.

4.2 Configuring MySQL Database:

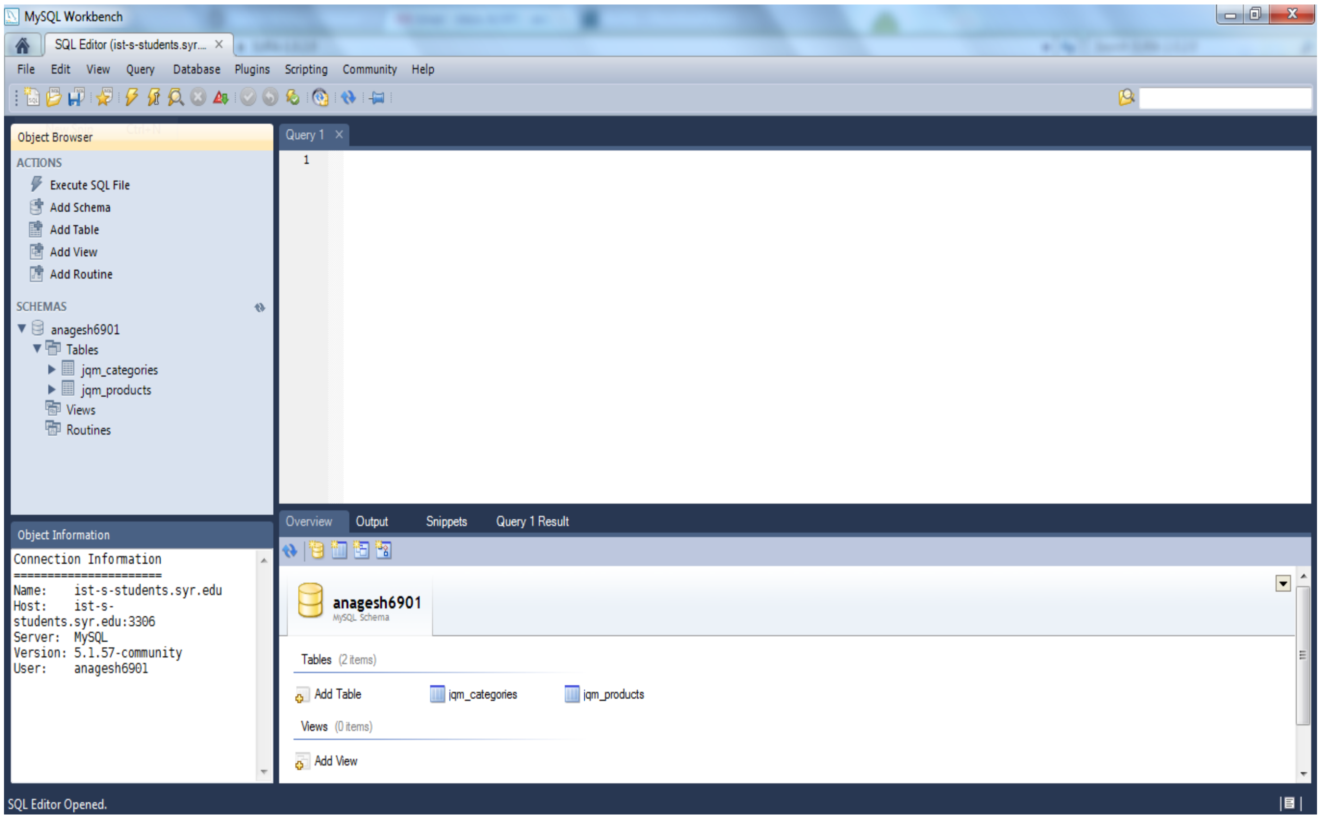
To configure and use MySQL database, follow the below instructions.

1. Download , Install and Configure MySQL Server by following the instructions present in http://dev.mysql.com/doc/refman/5.6/en/windows-installation.html
2. In this test, we had a MySQL server configured on a server (ist-s-students.syr.edu).
3. In order to use, the above MySQL server, download the MySQL workbench from http://dev.mysql.com/downloads/workbench/5.2.html. It is a GUI for MySQL servers using which we can easily design database schemas, execute queries and maintain the database.
4. Start the MySQL workbench and login with the credentials. Login with the username and password which are required to login to the MySQL server.


SQL Development
Connect to existing databases and run SQL Queries, SQL scripts, edit data and manage database objects.
Open Connection to Start Querying
Or click a DB connection to open the SQL Editor.
ist-s-students.syr.edu
User: anagesh6901 Host: ist-s-students.syr.edu:3306
New Connection
Add a new database connection for querying.

Server address: ist-s-students.syr.edu

5. Double click the connection and an editor window opens as shown below.



```
Connection Information
======================
Name:    ist-s-students.syr.edu
Host:    ist-s-
students.syr.edu:3306
Server:  MySQL
Version: 5.1.57-community
User:    anagesh6901
```

The editor has 4 major windows.

Object browser: In which the database schemas, tables, routines can be viewed.

Query Window: In which MySQL queries are executed.
Object Information: Which shows the server details
Overview-Output: In which we can see the output of queries executed, overall architecture of database.

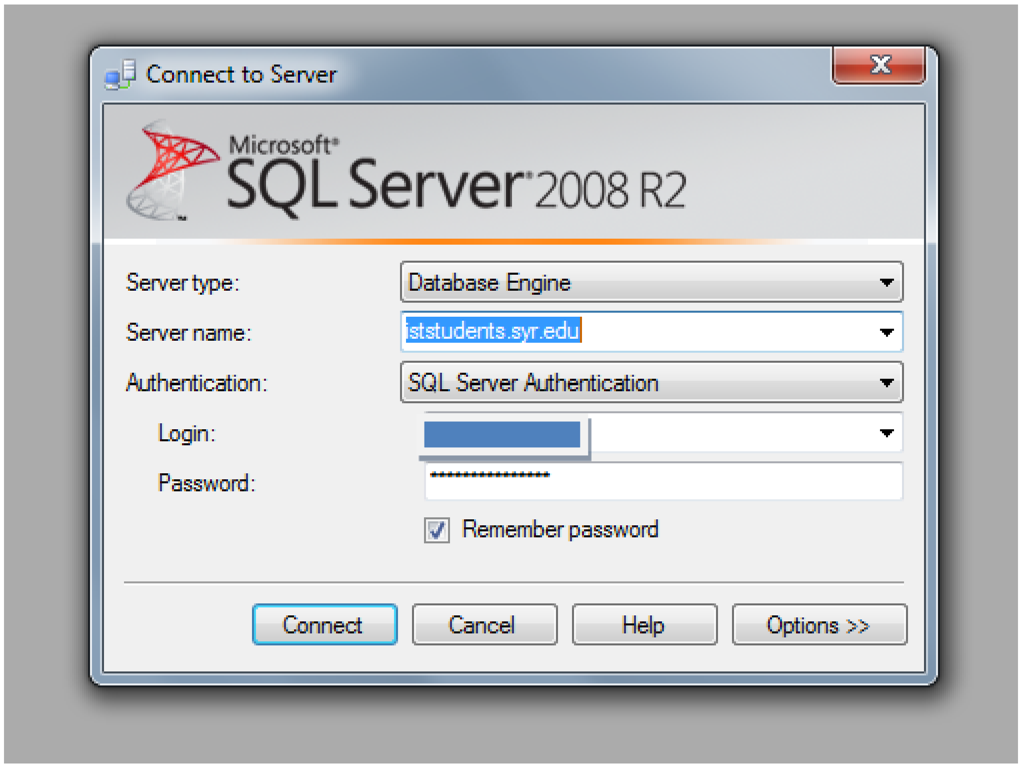
Example Query to create a table:

```
CREATE TABLE IF NOT EXISTS `jqm_categories1` (
        `id` int(6) unsigned NOT NULL AUTO_INCREMENT,
        `name` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
        `contains` int(6) NOT NULL,
       PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=4 ;
```

6. This finishes the configuration of MySQL database.

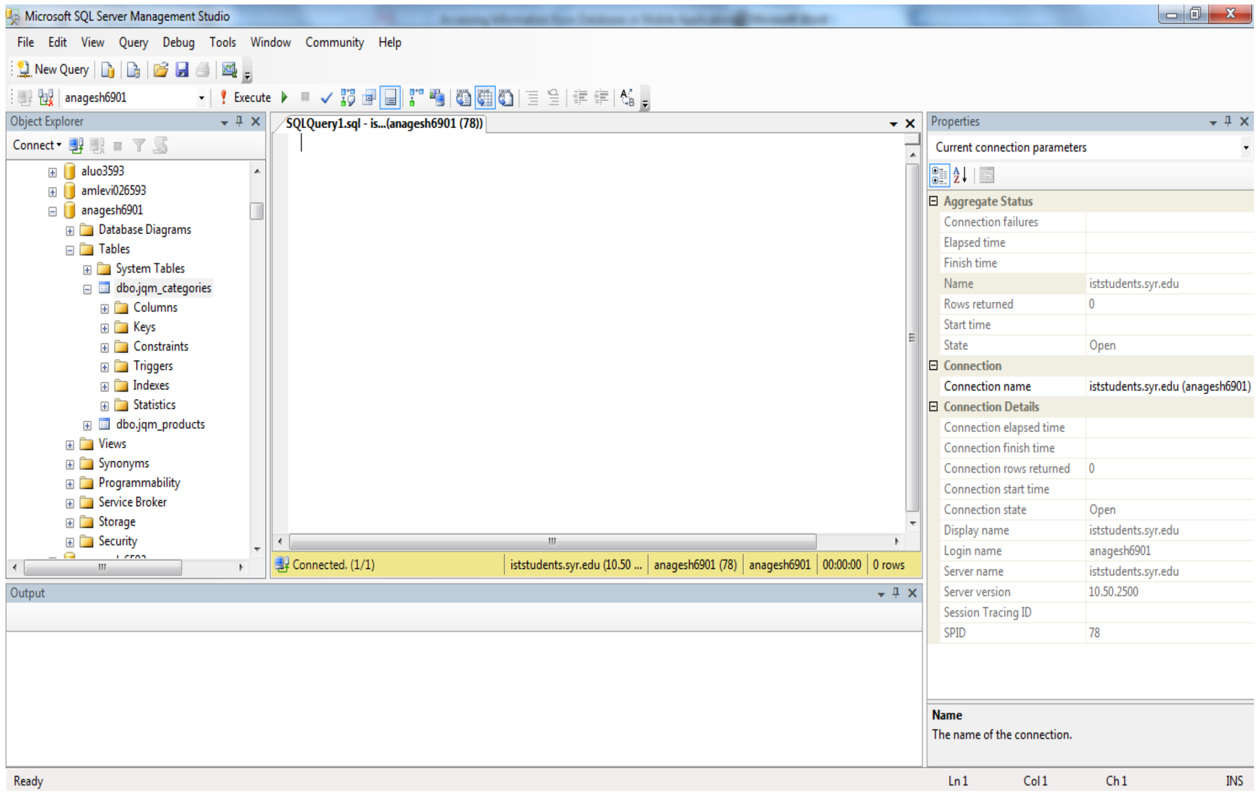
4.3 Configuration of MSSQL Database:

1. Download and Install MSSQL Server from Microsoft. In this test, the server is installed in ‘iststudents.syr.edu’.
2. Start the server from Start Menu. Enter the credentials in the dialog box displayed (Use credentials adequate to your case)



Server type: Database Engine
Server name: iststudents.syr.edu
Authentication: SQL Server Authentication
Login: [redacted]
Password: ****************
☑ Remember password

3. Once the connection is established, we can create databases, insert tables and add data to the same. The below image gives an overview of the startup window of the server.



Connected. (1/1) | iststudents.syr.edu (10.50 ... | anagesh6901 (78) | anagesh6901 | 00:00:00 | 0 rows

Aggregate Status	
Connection failures	
Elapsed time	
Finish time	
Name	iststudents.syr.edu
Rows returned	0
Start time	
State	Open
Connection	
Connection name	iststudents.syr.edu (anagesh6901)
Connection Details	
Connection elapsed time	
Connection finish time	
Connection rows returned	0
Connection start time	
Connection state	Open
Display name	iststudents.syr.edu
Login name	anagesh6901
Server name	iststudents.syr.edu
Server version	10.50.2500
Session Tracing ID	
SPID	78

Name
The name of the connection.

The window is very similar to MySQL server GUI, where we can run SQL queries, change the connection parameters and create database schemas.

4.4 Sample Query to create a table in MSSQL:

```
CREATE TABLE jqm_products (
        id INT NOT NULL IDENTITY(1,1),
        category INT NOT NULL,
         name VARCHAR(32)  NOT NULL,
        manufacturer VARCHAR(32)  NOT NULL,
        price INT NOT NULL,
         PRIMARY KEY (ID),
)
```

```
INSERT INTO jqm_products (category,name,manufacturer,price) VALUES
(1, 'MacBook Air', 'Apple', 999),
(1, 'MacBook Pro', 'Apple', 1500);
```

4.5 Sample Query to create a table in MySQL:

```
CREATE TABLE IF NOT EXISTS `jqm_products` (
        `id` int(6) unsigned NOT NULL AUTO_INCREMENT,
        `category` int(6) unsigned NOT NULL,
         `name` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
          `manufacturer` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
        `price` int(6) NOT NULL,
         PRIMARY KEY (`id`),
         KEY `category` (`category`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=12 ;

INSERT INTO `jqm_products` (`id`, `category`, `name`, `manufacturer`, `price`) VALUES
        (1, 1, 'MacBook Air', 'Apple', 999),
        (2, 1, 'MacBook Pro', 'Apple', 1500);
```

5 Sample PHP Proxy:

In this section, I will explain a simple example of a php file acting as a proxy between the client application and the backend database. It is responsible to receive inputs from the client application, communicate to the database, retrieve results and communicate back to the client application. Below is a simple code snippet of php file used with MySQL database.

```
<?php
        if (!$link = mysql_connect('ist-s-students.syr.edu', 'anagesh', '********'))
        {
           echo 'Could not connect to mysql';
           exit;                                                                  1
        }
        else
        {
        echo 'Connection succeded';
        }

        if (!mysql_select_db('anagesh_test', $link))
        {
           echo 'Could not select database';
                                                                                  2
           exit;
        }
```

```
    $sql    = 'SELECT * from jqm_categories';
    $result = array();
    $result = mysql_query($sql);
                                                                    3
    if (!$result) {
       echo "DB Error, could not query the database\n";
       echo 'MySQL Error: ' . mysql_error();
       exit;
    }

    while ($row = mysql_fetch_assoc($result)) {
       //echo $row['id'];                                           4
       echo json_encode($row);
    }

    mysql_free_result($result);
?>
```

The PHP proxy file contains 4 main sections as shown above.

1. In section 1, the PHP file needs to establish a connection with the remote database. The MySQL function used to connect to the database is
   mysql_connect ('ist-s-students.syr.edu', 'user_name', '*****')***
   3 parameters need to be passed to the function namely 'server address', 'user name', 'password'.

   '**echo**' is a php command to used to output the result onto the screen.
   '$**link**' is a variable which stores the result of the connection (Success or failure).

2. Once the connection to the database is established, a particular database has to be selected. MySQL has a function to accomplish the same.
   mysql_select_db('anagesh_test', $link)

   'anagesh_test' is the name of the database created in ist-s-students.syr.edu.
   '$link' is the value of the connection to the database.

3. After selecting the database, we need to execute the required query on the database, retrieve results to the php file and process them as JSON objects. This is done in section 3.

   $sql   = 'SELECT * from jqm_categories'; -> sql query to be executed on the database; which selects all the rows from the table 'jqm_categories'

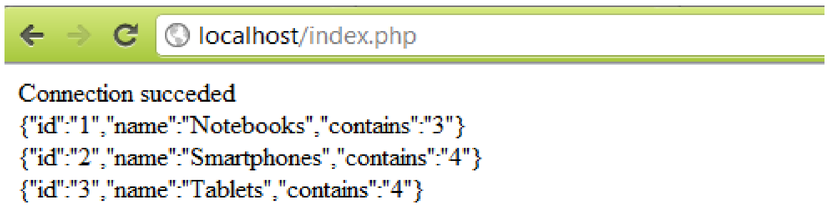
$result = mysql_query($sql); -> the result returned from the database is stored in an array called $result.

4. Once the results are obtained, we need to process the results to either JSON or XML format. We chose JSON as it is mobile application friendly, easy to parse and efficient.

   while ($row = mysql_fetch_assoc($result)) -> In this statement, the mysql_fetch_assoc function returns an associative array of the fetched rows and moves the internal data pointer ahead.

   echo json_encode($row);-> json_encode is a function which returns a JSON representation of the row/value. It is displayed on the screen using ‘echo’

5. The output is shown below.



```
Connection succeded
{"id":"1","name":"Notebooks","contains":"3"}
{"id":"2","name":"Smartphones","contains":"4"}
{"id":"3","name":"Tablets","contains":"4"}
```

6 Using Local Databases in HTML5:

One of the advantages of using HTML5 is, it allows us to make use of the local storage features of the browser. Most of the native browsers such as Google Chrome, Mozilla Firefox, Safari have the feature of local storage in which applications can store data either in the form of databases. The client side database storage API allows web applications to store structured data locally using a widely used medium- SQL. The API is asynchronous and uses callback functions to track the results of the database query.

The API to open the database looks like below.

var database = openDatabase("Database Name", "Database Version");

The above API call opens a database by specifying the parameters – Database Name and Database Version.

```
database.executeSql("SELECT * FROM test", function(result1) {
  // do something with the results
  database.executeSql("DROP TABLE test", function(result2) {
    // do some more stuff
    alert("My second database query finished executing!");
  });
});
```

The above API executes a SQL statement and returns the result in a callback function ‘result1’. The callback function can be used to execute any statements using the results retrieved. The local databases can be inspected using developer tools available in the webkit of the browsers. The below example shows creation of database, tables and method to insert data, drop data from tables in local databases. The database is inspected in the web inspector of chrome. The screenshots are also attached.

The main functions and API’s used for database creation are explained and can be found in the attached JavaScript file.

```
function initDatabase() {

            try {

                if (!window.openDatabase) {
                    alert('Local Databases are not supported by your browser. Please use a Webkit browser
for this demo');
                } else {
                    var shortName = 'DEMODB';
                    var version = '1.0';
                    var displayName = 'DEMODB Test';
                    var maxSize = 100000; // in bytes
                    DEMODB = openDatabase(shortName, version, displayName, maxSize);
                                  createTables();
                                  selectAll();
                }
```

- ‘*openDatabase(……)* is a function which opens the database.
- ‘*createTables()* ‘is a function where the tables are created. Its explained below.

CreateTables():

```
function createTables(){
            DEMODB.transaction(
        function (transaction) {
            transaction.executeSql('CREATE TABLE IF NOT EXISTS page_settings(id INTEGER NOT
NULL PRIMARY KEY, fname TEXT NOT NULL,bgcolor TEXT NOT NULL, font TEXT, favcar
TEXT);', [], nullDataHandler, errorHandler);
        }
    );
            prePopulate();
}
```

- ‘*transaction.executeSql’* is the API used to execute SQL statements. In the above statement, a table ‘page_settings’ is created if it is not present.
- ‘*prePopulate’* is a function called to insert data into the table created.

prePopulate():

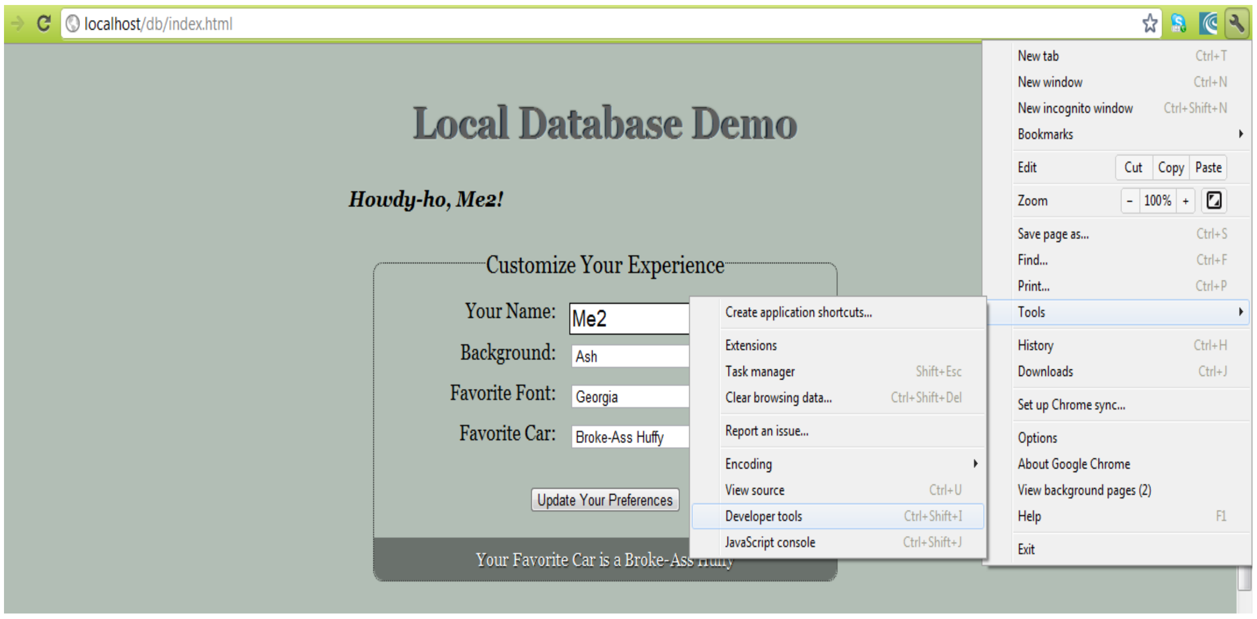
```
function prePopulate(){
        DEMODB.transaction(
          function (transaction) {
               //Starter data when page is initialized
               var data = ['1','none','#B3B4EF','Helvetica','Porsche 911 GT3'];

               transaction.executeSql("INSERT INTO page_settings(id, fname, bgcolor, font, favcar)
VALUES (?, ?, ?, ?, ?)", [data[0], data[1], data[2], data[3], data[4]]);
          }
        );}
```

In the above function, we insert values to the table ‘page_settings’ from the array of values stored in variable ‘data’.

Using Web Inspector in Chrome:

Local Storage like client side databases can be inspected in Chrome using web developer tools. In Chrome, click the Settings->Tools->Developer Tools as shown below.



localhost/db/index.html

Local Database Demo

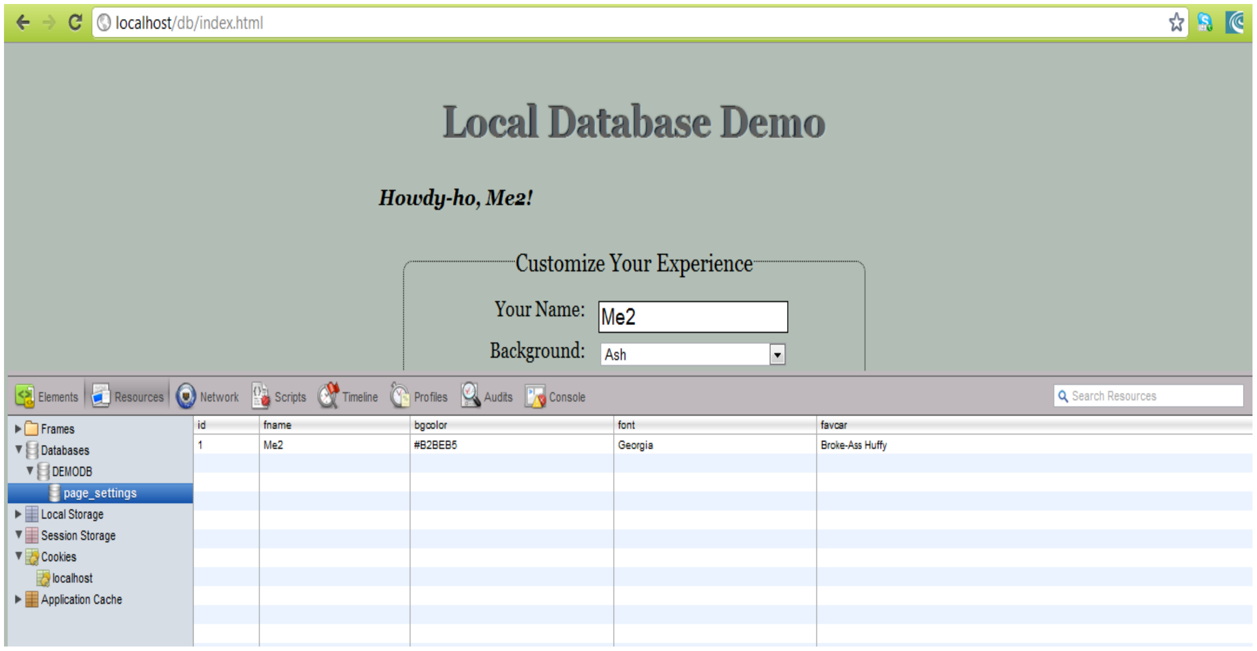
Howdy-ho, Me2!

Customize Your Experience

Your Name: Me2
Background: Ash
Favorite Font: Georgia
Favorite Car: Broke-Ass Huffy

Your Favorite Car is a Broke-Ass Huffy

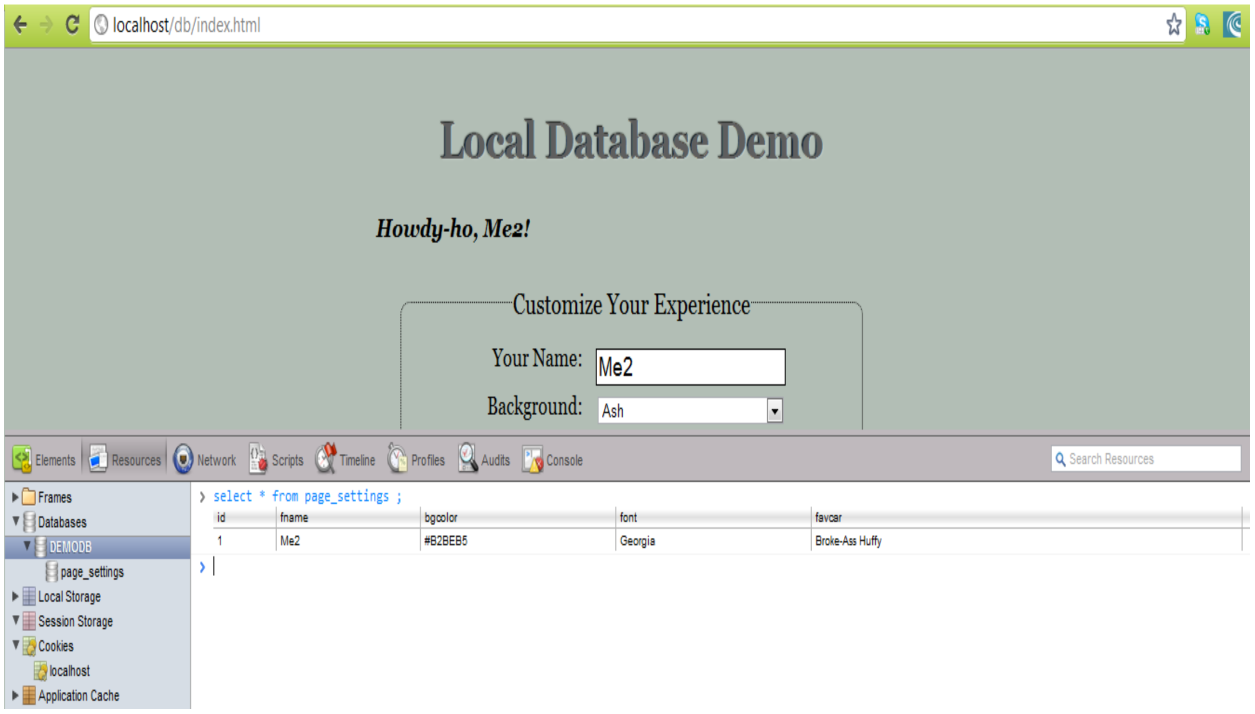
On the left hand side, under resources, select Databases->DemoDB->page-settings to view the database structure and also the tables with data on right hand side. A snapshot is provided below.



Your Name: Me2
Background: Ash

id	fname	bgcolor	font	favcar
1	Me2	#B2BEB5	Georgia	Broke-Ass Huffy

We can also SQL queries in the web inspector. Click on the database 'DEMODB'. On the right hand side, a window for running SQL queries is shown. Type any SQL queries to obtain the results instantaneously. A snapshot is given below.



Your Name: Me2
Background: Ash

```
> select * from page_settings ;
```

id	fname	bgcolor	font	favcar
1	Me2	#B2BEB5	Georgia	Broke-Ass Huffy

```
> |
```

7 Conclusion:

Databases are an integral part of many modern mobile applications, whether it is an internal database or external. Internal databases can be used to some extent where the stored data does not consume huge amount of internal memory of the mobile device. Storing data in a internal database also results in decreasing the response time of the application, memory leaks and application crashes. For mobile applications that deal with images, it is not recommended to use internal databases, cache to store images since it consumes large amounts of memory. Hence using external databases has become a very common approach to host data for mobile applications. This has many advantages such as zero maintenance of local databases, unlimited storage space to store both data and images and as a result fewer memory leaks and application crashes. But at the same time, mobile applications cannot access external databases directly by executing SQL queries. The MVC approach discussed in this paper is useful in this situation and we hope that the guidelines and examples provided can be useful to other developers.