International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 3, Issue 1, Jan 2014, 1-6

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

DISTRIBUTED CONNECTIVITY OF MOBILE APPS WITH MAINFRAME APPS

TANUL BHASIN[1] & SAGAR GUPTA[2]

[1]Systems Engineer, Infosys Limited, Jaipur, Rajasthan, India

[2]Systems Engineer, Infosys Limited, Bangalore, Karnataka, India

ABSTRACT

In today's advancing technology it has become difficult for the latest technologies to use legacy applications for data retrieval or for business invocation. With mobile phones having continuously updating their operating systems (OS) which are based either on windows, IOS or Android, old technology such as mainframe is becoming obsolete. Today's current technology demands the usage of legacy application with distributed connectivity so other frameworks can also use the business modules without re-writing them again. This research describes how to create an Android application and use it to connect to DB2 running on mainframe as the server side database and to upload or to retrieve data in a DB2 based system which can be used as a base to expose DB2 sub-system of Mainframe server for distributed client access for web or mobile apps.

KEYWORDS: Android, Mainframe, JAVA Servlet, Distributed Connectivity, Android Connectivity with Legacy Application

INTRODUCTION

Clients using mainframe application find it difficult to remotely access their data which is stored on a legacy application. The problem can be resolved by creating an application which can be accessed remotely using a smart phone and can be based on Android, IOS or windows, which can be used to upload and download data from the legacy applications. For further discussion Android will be referred to as client OS. Android, a joint venture from GOOGLE and OHA (Open Handset Alliance) is the platform to create applications for mobile phones, the app provides an interface for the mainframe application, J2EE acts as the controller and the COBOL application code which is an interface to the web service is the model in the mainframe. A simple application written in COBOL fetches the user information from DB2 tables and can be scaled up to be invoked from a mobile phone aided with an Android application. For this purpose, the application is exposed as web service which can be invoked from any distant environment. J2EE acts as the distributed client interfaces between the Android app and the Mainframe Application

The android application here uses DB2 running on mainframe to store and to retrieve BLOB (binary large object or basic large object) objects. The connectivity between DB2 and Android is established by using an in-lined layer programmed in JAVA Servlet, which aid the program to accept the incoming request sent from Android app in form of binary stream and then convert it to BLOB object to insert it into DB2 table; and similarly sending response in form of image back to Android app after converting the fetched BLOB object from DB2.

BLOB objects are the chunks of binary data stored on a single entity present in a database management system. Blob can be of many types like images, audio or multimedia objects etc. A binary executable code can also be stored as BLOB. Using these objects large amount of data can be transferred from one database to another with proper filtering and error correction

Character Large object (CLOB) also stores the collection of data in the form of characters in database management system can also be used instead of BLOB. In this the data can be converted internally for storage, based on the character set of database, which would then be reconverted to the proper set on selection later. The challenge with CLOB occurs when, data other than character data is stored in one operating system and is retrieve on another operating system, as the result might not necessarily be the same. Whereas BLOB stores data in binary format, therefore any information in BLOB is not converted at all and the order of bits fed into a BLOB remains consistent. The other major reason for using BLOB is data compression and a compressed object in a BLOB is not manipulated. Whereas, CLOB gives various issues with the compressed objects

The mainframe application is exposed as web-service which can be invoked from any distributed environment. The web service is created using WSDL (Web Service Definition Language) which is independent of browser and upon invocation exposes the business logic, which easily provides the result according to the business logic. With a web service hosted by the application it becomes easy for any mobile or browser to access the application. The Android application empowered with J2EE acts as a distributed client interface between Android App and Mainframe, and with the help of ISO, HTML, Java, C#, it makes the business logic platform independent.

A known problem with the android application is its features of 6 sec response time from the application, if the main thread of the application is unable to respond back in 6 sec android OS will force close the application and kills all the threads created by the application. To avoid this, the application has to be developed Light weight model for server communication.

MATERIALS AND METHOD USED

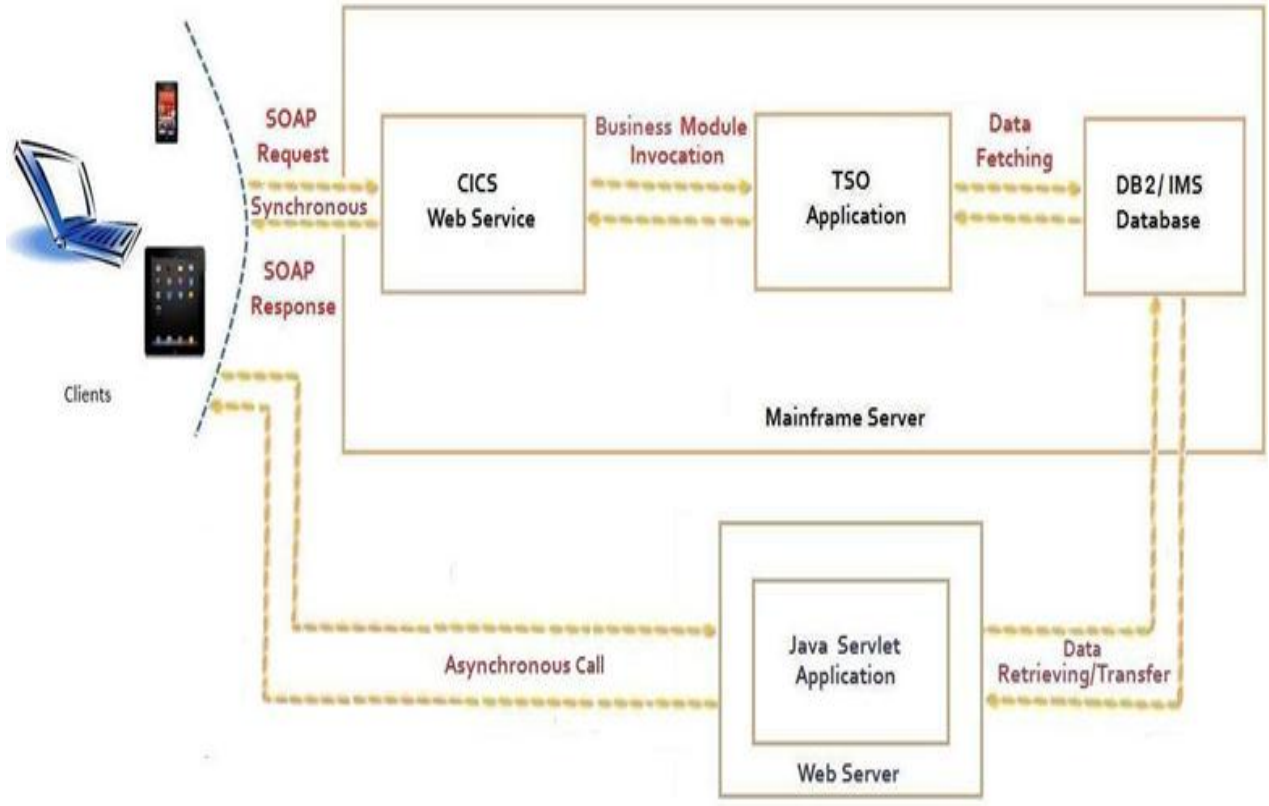

Figure 1: Client-Server Communication

Clients: It is an application working on an android environment which is further used to connect to a legacy application. This application makes synchronous and asynchronous calls.

CICS Web Service: Web-services can be used in CICS sub-system to expose the business logic for distributed environment, mainframe server can only support only SOAP web-service and HTTP/HTTPS (based on the CICS version) can be used as protocol for connectivity. Web-service can be used in mainframe in following three ways:

- **Bottom-Up-Approach:** In which mainframe will act as web-service provider.
- **Top-down-Approach:** In which mainframe will act as web-service invoker.
- **Meet-In-Middle:** It's a hybrid approach where a wrapper program is used that maps between the data format generated by the invoker program in mainframe sub-system and format used by CICS application program.

TSO Application: The application program through which the business is implemented, TSO application can be written in COBOL, PL/I, C, C++ or Assemble language which has a communication module with CICS, the data transfer between CICS and TSO application can be done in two ways:

- **Using Common Area:** Every transaction in CICS has a common area where the participating modules can save the data temporarily, when the module is exposed as web-service this area can be used to send/receive inputs from the client application.
- **Using Channel-Container Architecture:** The main disadvantage with CICS common area is its max length for data storage which is 32K, so if the input or the output data is more than 32K channel-container serves the purpose.

Database: An enterprise can choose between DB2 or IMS as their database based on their business requirements, both having their own advantages and disadvantages. IMS is a hierarchical database in which information is structured in the form of records. These records are further subdivided into a hierarchy of related segments. The meaning of hierarchical itself defines that there are levels of data and as we progress from level down, more and more information related to the general information at the top level is given. DB2 is a relational database management system and is used to run on multiple virtual storage mainframe platforms and allows to create, store, update and retrieve data in systematic manner. It is mainly used to shift from the prevalent hierarchical database to new relational model.

It can serve to a number of difference operating system platforms. DB2 was initially designed to work on mainframes however today it can be used in various platforms like UNIX, windows and linux along with others. DB2 presents the logical and structured view of data due to which user need not to worry how the data will physically store. It also provides the layer of Independence due to which when the structure of data changes there is no need to re-write whole COBOL program again. XML data is implemented natively in DB2 due to which data can be accessed faster using XQuery.

It is possible to use any of the DB2 or IMS database, however, there are some requirements due to which DB2 is more preferable over IMS like it is easy to use cursors in DB2. Also it is possible to write SQL queries in interactive tool of DB2 (SPUF1) to perform operations on DB2. But it is undeniable that DB2 will consume more CPU than IMS. DB2 optimizes the queries internally whereas IMS developers makes access path to data. However, this additional requirement of CPU make the DB2 more beneficial because it leads to enormous benefit of database optimization and ad hoc query support is also better.

DISCUSSIONS

To establish the connectivity of the android application with the mainframe a synchronous or asynchronous call

can be made which makes the application lighter. The android application should have necessary Internet access permission, as remote procedure calls are only possible through Internet.

Prerequisites for the call are:

- Mainframe Z/OS
- Android phone/emulator
- JDK 1.5 or above
- J2EE Webserver
- Android SDK
- CICS 3.2
- DB II or IMS

The android application will always make a REST based call, but the mainframe does not support REST based calls and supports only SOAP based requests.

REST (Representational State Transfer): With this the services can be interacted via stateless representations of the targets of the service. Rest based URI's is specified as:

https://domainname.com/service/method?parameter=var1&parameter=var2

Rest based services are stateless because http/https are natively stateless. It uses the security of the transport layer that was implemented to support world-wide web. It is built on top of HTTP operations: GET, PUT, POST, and DELETE. This security is based on two areas:

- End to End security, which is provided by Secure Socket layer
- Authentication ranges from basic HTTP authentication to providers like OAUTH2

In REST based calls data is transferred in JSON (JavaScript object notation) format. JSON is derived from java script language which is used to represent simple data structures and associative arrays, called objects.

SOAP (Simple Object Access Protocol

It is a protocol which is specified for exchanging structured information in XML format in the implementation of Web Services. It is a well-known technology that has been used effectively in SOA framework. SOAP was defined to take advantage of various transport layers like synchronous HTTP/HTTPS, asynchronous queues (MQSeries) and even over email. This makes SOAP as a single solution for various interconnectivity problems.

So to make a connection of Android with mainframe, a Java servlet is required. Servlet are a type of JAVA programming technique which are used to extend capabilities of server so that the servers' innovation can be done by host application running on remote machine via request-response programming model. Client's android application will make restful web-service call to Java Servlet. This servlet can make both rest and soap based calls, so when servlet takes a REST based input it further makes a soap based call to mainframe which leads to transfer of data between android and mainframe. The servlet is deployed on Web Server. This system will connect to J2EE server through the libraries which were present in 'classpath' of the project.

On the basis of requirement the user can make synchronous or asynchronous calls. If user is sending some data and needs an immediate response then it is required for user to make synchronous calls. However, if user wants to send or receive large files like in uploading and downloading of images or music files then it is required to make asynchronous calls. The reason to make asynchronous calls is to avoid closing the data connection when receiving heavy data files, as the Android Application force closes the application if the OS will not receive response from the main thread of the application within 6 seconds.

Synchronous Calls

Here a 3-tier model is used for developing android application. In which the first layer is UI (User-Interface) layer and is designed in normal android platform, the interlaying layer which is a controlling and interfacing layer is programmed in JAVA which will take the inputs from the android app and format the inputs so that data can be sent to mainframe, which then invokes the web-service and after receiving response from the web-service and sends the result in Android compatible format to Android application.

The Android – Mainframe framework is organized around the common Model-View-Controller pattern.

Model: The model represents data or data container. The mainframe application will get data from database DB2 based on the input received and send a response back. This mainframe application is exposed as CICS web service for sending text format data.

View: The View is the portion of the application responsible for the display. The android app is the view of this architecture. This is where the input to the model and the response comes from the model is displayed to the user. User will always interact with this layer. User's actions on this layer triggers application functions

Controller: The J2EE controller responds to all the external actions. On the user's actions, the control is passed to the controller which will take care of the logic to invoke the mainframe web service and also interface with the view. This will take care of the logic to prepare the model to be sent to the view.

Asynchronous Calls

It is 3-tier architecture, in which the Android Application will communicate with DB2 through mid-level layer which will act as interface between Android Application and DB2 database to convert the data sent from or to Android app into required format without using web-services. This common scenario of image uploading/downloading in discussed further; In the case of sending data to the server where the transmission may exceed the cap of 6 seconds asynchronous calls should be used, in asynchronous calls main thread will create a sub-thread which will take care of data transmission between device and server making the main thread free for further usage. Once the sub-thread finishes the transmission it will notify the main thread about the status of the transmission, upon which main-thread will work.

So while sending heavy data like images or documents from the device to DB2 sub-system asynchronous calls are the most effective techniques, in which the client application will create an asynchronous task which will send/receive the data from the DB2 sub-system and will notify once the transmission finishes.

As Asynchronous calls are not supported by web-service architecture, a mid-layer of J2EE application is used which will receive the input from the invoking client application and process the request with the DB2 sub-system and receive/send the data to the database. When the transmission is complete it will notify the client application about the status of the transmission. Data transmission can be done in many ways but the best and safest method is to use JSON transmission.

BENEFITS

Functionality Compliance

The android application in a distributed environment can invoke a mainframe legacy application through the interface of the J2EE server.

Scalability

An emulator referred to the approach is a virtual mobile. All the functionality implemented through the emulator will work the same even when tested through a physical mobile phone supported with android.

The above discussed approach is implemented using Android version 2.3.3 (GINGERBREAD) as reference. The same can be implemented on the Version 4(ICE CREAM SANDWHICH) and further.

Performance Compliance

Android application, receives input from the user, sends the request to the mainframe application and waits for a response, only for 6 sec. If there is no response sent from the other end, the android application will be force closed.

This performance implication is because of a SYNC CALL made to the server in which the main activity will wait for the response. This can be overcome by implementing an ASYNC CALL which will free the main activity thread under held for response. Once the response is received, then main activity thread is resumed.

CONCLUSIONS

The input for the mainframe application is given through the Android application and the response from the application is again seen on the android application. The mainframe application is exposed as a web service which is invoked by the android application through the J2EE server thereby making it easy to upload and download heavy files also.

Image Uploading: The image was sent from Android app to sever-side application and inserted into DB2 table.

Image Downloading: Requested image was fetched out from DB2 from table based on image id send from app to sever application and the image was downloaded asynchronously and shown on the screen.

REFERENCES

1. Web Services An Introduction, 2nd Edition, BV Kumar, SVS.
2. IBM DB2 9.7 Advanced Administration Cookbook.
3. Professional Android 4 Application Development.
4. Android for Programmers: An App-Driven Approach (Deitel Developer Series).
5. Redbook: Java Stand-alone Applications on z/OS Volume II.
6. Redbook: DB2 Version 9.1 for z/OS Application Programming Guide and Reference for Java - SC18-9842-11.