In this two-part tutorial series, learn how to design and build messaging service applications. Using a Liberty for Java application as an example, you'll see how to integrate two external messaging services using their APIs, IBM Bluemix SQL Database Service for persistence and Dojo JavaScript framework to provide a rich Web 2.0 front end. In Part 1, we looked at the different components of an existing multimessage application. In Part 2, we focus on deploying the application on Bluemix, on external messaging services API account details, and on testing locally on IBM WebSphere Application Server V8.5 Liberty profile.

In Part 1 of this two-part tutorial series, covered the application architecture, I listed what you need to build a similar app, and went through the steps to obtain the example application code and set up your development environment. I also covered the design and implementation of our rich three-tier multimessage web application by going through each tier and providing important code snippets.

In Part 2, we'll walk through the multimessage app deployment details and test locally on IBM WebSphere Application Server V8.5 Liberty Profile. You'll also learn more about external messaging services.

If you haven't yet registered for Bluemix, gotten the sample code, and set up your application, do so by reading Part 1.

To deploy our example application on Bluemix, perform the following steps:

**Step 1. Create a SQL DB Service on Bluemix**

1. Log in to **Bluemix**. On the top right header, click **Catalog**. You'll be taken to a new page. In the left navigation bar, under Services, click **Data & Analytics**.
2.

Under Data & Analytics, click SQL Database.
3. On the Add Service dialog, select Leave unbound in the App field. Enter a service name of your choice (no spaces). For now, select Free Beta for the Selected Plan. Click CREATE.

Step 2. Create Nexmo and Twilio accounts

To send SMS messages, you first need to create your messaging services accounts on Nexmo and Twilio. The process is essentially the same for both, except Twilio requires an additional step of obtaining an SMS-capable number.

Nexmo

1. Create your account on Nexmo. Keep a record of the phone number you provide when creating the account. This number is automatically added as a test number. In later steps, you will use this number as NEXMO_SMS_FROM.
2. Log in to Nexmo and click API Settings.
3. Obtain the API key and secret.
4. Optionally, you can add more test numbers by clicking on API Settings > Edit > Test Phone Numbers. For the purpose of this article, one number is fine.

Twilio

1. Create your account on Twilio and log in.
2. Click on **Numbers > Buy a Number**.

3. Choose your country, select **SMS** for Capabilities, and click **Search**.

![Buy A Number](image)

4. Choose one number from the given list of available numbers. Make sure to select an SMS-capable number. You will use this number as **TWILIO_SMS_FROM** in later sections.

5. From the login Profile drop-down, click **Account**.

6. Make a note of the **AccountSID** and **AuthToken** of the live API. Click on **Lock** to see the AuthToken.

**Note**: To avoid future charges, release your number at the end of this tutorial unless you plan to continue using it.

### Step 3. Publish the app to Bluemix from Eclipse IDE

In **Part 1** of this tutorial series, you installed the Bluemix tools in Eclipse and added Bluemix server to the Eclipse Servers list. Now you'll set up Bluemix server in Eclipse and publish your app.

1. From your Eclipse IDE, open the Servers tab and click **Windows > Show View > Others > Server > Servers**.

2. Double-click on the Bluemix server and validate your account. Keep the defaults for the rest of the configurations.

3. Start the server if it's not already started. Wait for the server status to indicate that it is started. Don't restart your server if it's already started, as this would take a long time.

4. Add your application to the server by right-clicking **IBM Bluemix Server**. Select **Add and Remove > Add multimessage > Finish**.

5. You will be prompted to provide a unique name for the application. That name must not be "multimessage." Check **Save to manifest file** and click **Next**.
6. On the Launch Deployment panel, click **Next**.

![Launch Deployment Panel](image)

7. When given the option to choose the service to bind to the application, select the SQL DB service you created, and click **Next**.

8. The Environment Variables panel contains the environment variables and their values. You can adjust the environment variables for your application.

![Environment Variables Panel](image)

9. Using the Edit Variable Entry dialog shown above, enter the following list of environment variables and their values:

![Edit Variable Entry Dialog](image)
Listing 1. Environment variables

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEXMO_SMS_FROM</td>
<td>Obtained in step 2 &gt; Nexmo &gt; 1. Example value: 353862335871 (must not prepend + or 00 sign)</td>
<td></td>
</tr>
<tr>
<td>NEXMO_API_KEY</td>
<td>Obtained in step 2 &gt; Nexmo &gt; 3. Example value: 12345678</td>
<td></td>
</tr>
<tr>
<td>NEXMO_API_SECRET</td>
<td>Obtained in step 2 &gt; Nexmo &gt; 3. Example value: 12345678</td>
<td></td>
</tr>
<tr>
<td>TWILIO_SMS_FROM</td>
<td>Obtained in step 2 &gt; Twilio &gt; 4. Example value: +353768886708 (must prepend a + sign and no spaces in between)</td>
<td></td>
</tr>
<tr>
<td>TWILIO_ACCOUNT_SID</td>
<td>Obtained in step 2 &gt; Twilio &gt; 6. Example value: ACe69a03aacff17c501eb1da1e5ee456ee</td>
<td></td>
</tr>
<tr>
<td>TWILIO_AUTH_TOKEN</td>
<td>Obtained in step 2 &gt; Twilio &gt; 6. Example value: ACe69a03aacff17c501eb1da1e5ee456ee</td>
<td></td>
</tr>
</tbody>
</table>

10. Click **Finish**. Your application will now start to publish.

11. You can view the application logs in the Console tab of Eclipse IDE. You can also see the logs on **Bluemix**. Open your application on Bluemix and select **Files and logs > Logs > messages.log**. If the application is not behaving as expected, it’s critical that you keep an eye on the logs.

If there are no errors in the log, your application is successfully deployed on Bluemix!

12. If you get the error **Error: Client error - Error performing operation: 500 Error** at this stage, it means that the Bluemix service is temporarily down. For details, refer to **500 Error with Push**.

If that happens, wait for the service to come back up, then remove your application from Eclipse IDE. To do this, go to the Servers tab and right-click **Bluemix server > Remove > Do not remove the DB services during this wizard > Finish**.

Add the application again following Steps 3.4 to 3.8 above. Because you chose to save to manifest file, you don't need to provide the environment variable values again.

13. View your application's status by opening your Dashboard on Bluemix, clicking the name of your space, and selecting **CF Apps**, as follows:

14. Click on the application to see more details. IBM Bluemix enables your application to deploy in minutes and gives you the ability to adjust the resources to meet your application's
demand. For example, you can increase the number of instances and memory at any time.
15. In the SQL Database section, click the **Show Credentials** link and make a note of the following details. These details are required to connect to the database later.

![Show Credentials](image)

16. The following fields are particularly important to note:
   - **name**: This becomes part of the JNDI service name while configuring the environment for local testing.
   - **port**: The DB port
   - **db**: The database name in db2 client
   - **username**: Used for authentication and also as a schema name
   - **hostname**: This becomes `servername` when configuring the environment for local testing
   - **password**: Password for authentication

**Step 4. Create a database table and insert sample contact data**

To create a database table and insert sample data for contacts, you can use either IBM Data Studio Client or IBM SQL DB Service web console.

**Option 1. Create a database table and insert contact data using IBM Data Studio Client**

1. To use IBM Data Studio, follow the [Download and install IBM Data Studio](#) guide.
2. Use the `create_tables.sql` and `insert_sample_data.sql` files provided with the application code to create the table and insert contacts respectively on your IBM SQL DB service.
3. Update the schema name `USER03749` in both files to the one you obtained in Step 3.15 above.
   - **Note**: Schema name is the same as `username`. 
4. Connect to your SQL DB Service from Data Studio Client. Open the Data Source Explorer and click **View > Database connections > New**. Provide the connection details of your SQL DB service. Click **Finish**.

![New Connection dialog box](image)

5. Open `create_tables.sql` by opening the Data Studio Client IDE and clicking **File > Open > <path to the create_tables.sql> > OK**.

6. Click the **Run Sql** button at the top right. This creates the `ContactList` table.

7. Once the table is created, repeat Steps 5 and 6 above for `insert_sample_data.sql` to populate contacts.
   
   **Note:** The sample `insert_sample.sql` file already contains some contacts. Update the `contactName` and `contactNumber` fields with your own data.

8. Run the SQL to insert the data.
   
   **Note:** Do not change the `userid` field. For now, keep it `user1`. Check out the **next steps** at the end of the tutorial for details about future enhancements you can do.
Option 2: Create a database table and insert contact data using IBM SQL DB Service web console

If you are using the IBM SQL DB Service web console, follow these steps:

1. Go to your space on Bluemix. Under SERVICES, click SQL DB Service. In the upper left, select Launch. On the new page, click the Manage tab > Work with Tables > plus sign (+)

![Create, drop, and work with tables](image)

2. In the SQL editor, insert the following SQL statement and click Run DDL. Note: Sometimes the schema is not shown in the above page. Ignore this issue. Once you run the following command, the schema automatically appears.

**Listing 2. CREATE TABLE ContactList**

```sql
CREATE TABLE ContactList (
    Id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
    ( START WITH 1 INCREMENT BY 1 MINVALUE -2147483648
    MAXVALUE 2147483647 NO CYCLE CACHE 20 NO ORDER ),
    UserId VARCHAR(128) NOT NULL,
    ContactName VARCHAR(128) NOT NULL,
    ContactNumber VARCHAR(128)
) DATA CAPTURE NONE COMRESS NO;

ALTER TABLE ContactList
ADD CONSTRAINT ContactList_PK PRIMARY KEY
(Id);
```

3. Click the plus sign (+) again. In the SQL editor, insert the following SQL statement and click Run DDL. You can modify the contacts with contacts from your preferred list. You can also add more insert statements like these.

**Listing 3. SQL insert statement**

```sql
insert into CONTACTLIST (USERID, CONTACTNAME, CONTACTNUMBER) values ('user1', 'Ahmad', '+353862335871');
insert into CONTACTLIST (USERID, CONTACTNAME, CONTACTNUMBER) values ('user1', 'Roger', '+353861241422');
```
**Note:** Do not change the `userid` field. For now, keep it `user1` for simplicity. Check out the next steps at the end of the tutorial for details about future enhancements.

4. Verify that the data is inserted. On the same page, click on your schema and select CONTACTLIST table > Browse Data.

### Step 5. Review environment variables

Bluemix provides two types of environment variables, VCAP_SERVICES and USER-DEFINED.

VCAP_SERVICES contains connection details of the Bluemix bindable services, SQL DB Service in our case. To get VCAP_SERVICES, log in to Bluemix, go to your Dashboard and your space. Click CF Apps > Your app > Environment Variables > VCAP_SERVICES.

USER-DEFINED contains user-defined variables. Here you can see the variables you provided when you published your app from Eclipse IDE.

![Environment Variables](image)

The application is now ready to use. Open a new browser window and enter the URL for your application: http://<your route>/index.html. For example, the route to the sample application is http://multimessage.mybluemix.net/index.html.

Obtain your route from your application in the Bluemix Dashboard, as shown below.

![Application Routes](image)

**Note:** If there are errors in deployment or other issues with the application, these can be diagnosed by deploying locally as explained in **Step 7** below.
Step 6. Review multimessage app running examples

1. Here's an example of a text message that was sent successfully:

   Select Messaging Service:
   - Nexmo
   - Twilio

   Select a Contact to Send SMS to:
   Message:
   Ahmad - +353862335871

   hello SMS!

   Send SMS  Reset

   Message service used: nexmo - status: Message successfully delivered.
   Message price: 0.03500000 Remaining balance: 1.82500000
2. When an environment variable is incorrect, it may result in an error, like the following one. In this case, the TWILIO_SMS_FROM value is wrong. I used a phone number that was not SMS enabled.

![Select Messaging Service]

Step 7. Test application on local host

The following steps show you how to set up and deploy your application locally on WebSphere Application Server (WAS) V8.5 Liberty Profile.

1. If you followed Part 1 of this series, your already added your application to the local WAS server (see Step 2 in Part 1).
2. Double-click on **WebSphere Application Server V8.5 Liberty Profile at localhost**, which gives you the following configuration:

![WebSphere Application Server V8.5 Liberty Profile at localhost](image)

**Overview**

### General Information
Specify the host name and other common settings.

- **Server name:** WebSphere Application Server V8.5 Liberty Profile at localhost
- **Host name:** localhost

### Liberty Profile Settings
Specify server configuration and publishing behaviour.

- **Liberty profile server:** defaultServer

Open server configuration

- **Run applications directly from the workspace**
- **Stop server on workbench shutdown**

3. Click on **Open server configuration > Source tab**.

4. In the editor, add the following code snippet above the `webApplication` element to configure `dataSource`.

### Listing 4. Data source configuration

```xml
<jdbcDriver id="DB2">
  <library name="Db2library">
    <fileset dir="C:\software\db2drivers\10.5"/>
  </library>
</jdbcDriver>
<dataSource id="SQLDB" jdbcDriverRef="DB2" jndiName="jdbc/idea1-sqldb">
  <properties>
    db2.jcc databaseName="SQLDB" password="123456" serverName="75.126.155.153" user="user03749"/>
  </properties>
</dataSource>
```

5. Update the following fields in the above snippet with the credentials for your app. Refer to your DB Service details displayed on the Show Credentials screen in Step 3.15 of this article for your app details. Here's how they map to the above fields:

- **jndiName:** "name" prepended with `jdbc/`.

• serverName: "hostname"
• user: "username"
• password: "password"
• databaseName: "db"
• DB2library: See next step below.

6. Db2library refers to the folder containing the DB2 driver, db2jcc.jar. Provide the path to its location in your environment. Obtain this JAR as follows:
   a. While creating your schema and inserting contacts data, if you installed IBM Data Studio Client, obtain the db2cc.jar file from the path <IBM Data Studio Client installation>/DS4.1.0\dsdev\jar.
   b. If you are not using IBM Data Studio Client, you can obtain db2cc.jar by following these steps:
      • Go to DB2 JDBC Driver Versions and Downloads. Click on v10.5 FP5 > 3.69.24 from the DB2 Version 10.5 table.
      • Download the appropriate pack for your environment. For my Windows environment, I use DSClients--jdbc_sqlj-10.5.0.5-FP005 > v10.5fp5 jdbc_sqlj.tar.gz.
      • Save v10.5fp5 jdbc_sqlj\jdbc_sqlj\db2_db2driver_for jdbc_sqlj\db2jcc.jar from this archive.

7. To test the application locally, you need to update the following URLs in the source code using Eclipse IDE. These URLs are located in <your project>/WebContent/js/multimessage/MessageForm.js:
   a. Change /api/contacts/userid/user1 TO:/multimessage/api/contacts/userid/user1
   b. Change /api/messageservice/sendsms TO:/multimessage/api/messageservice/sendsms


Listing 5. Nexmo key and secret

    public static String sendSms(String contactNumberTo, String smsText) {
        Account account = EnvUtils.getcredentials("NEXMO_API_KEY", "NEXMO_API_SECRET","12345678",
            "12345678");
    }


    return (smsFromNumber == null) ? "353862335871" : smsFromNumber;

10. Update smsFromNumber for Twilio in the <your project>/src/com.multimessage.messageservice.twilio.TwilioMessageSend.java file.

    params.add(new BasicNameValuePair("From",
        (smsFromNumber == null) ? ",353861800384" : smsFromNumber));


    Account twilioAccountCredentials = getcredentials
        ("TWILIO_ACCOUNT_SID",
        "TWILIO_AUTH_TOKEN", "AC34567890123456789012345678901234",
        "1234567890123456789012345678901234");
12. Update the jndiName field value (idea1-sqldb) and the username field value (user03749) with your Bluemix Database Service values in <your project>/src/com.multimessage.db.DbLayer.java.

Listing 6. Update JNDI name

```java
private static String getDataSourceName() {
    String dataSourceName = EnvUtils.getDataSourceName();
    if (dataSourceName == null) {
        return "jdbc/idea1-sqldb"; //update value after /
    }
    return dataSourceName;
}

public static String getSchemaName() {
    String schema = EnvUtils.getSchemaName();
    if (schema == null) {
        return "user03749"; //update this value
    }
    return schema;
}
```

13. Make sure the above changes are saved, then right-click WebSphere Application Server V8.5 Liberty Profile to start the app. If it's already started, you need to restart at this time.

14. Once the application is started, copy the URL (http://localhost:<server port>/multimessage/) into a browser. You can now use the application locally like you were using it on the Bluemix Server. You can also get the local URL in the log records under the Console tab of Eclipse IDE.

Troubleshooting tips

- After several restarts of the server or removing/adding the app, you may begin seeing the error ClassNotFoundException. For help, refer to this dW Answers known issue and solution. It worked for me.
- If you change the configuration or app code and it is not being applied, follow these steps:
  1. Go to Eclipse IDE and click Project > Clean. Select the project to clean and click OK.
  2. Delete the browser cache and refresh the browser page.
- Refer to logs in the Eclipse IDE Console tab when debugging.
- Debug from the front end using advanced tools like Chrome DevTools or Firefox Firebug. Chrome DevTools are available out of the box.
- On the page you want to debug, press F12, and refer to the Console and Network tabs.

Conclusion and next steps

In this two-part tutorial series, I demonstrated how to design and build messaging service applications, using a Liberty for Java application as an example of how to integrate two external messaging services using their APIs, IBM Bluemix SQL Database Service, and Dojo JavaScript framework. We examined an existing multimessage application, from high-level design to low-level code details, and walked through the deployment details of the application on Bluemix, as well as external messaging services API account details, and testing locally on IBM WebSphere Application Server V8.5 Liberty profile. You now have all the information you need to create your own messaging service application. If you performed the steps with me as you read these articles, you already have one.
Next steps

Next steps for this application include:

- Provide a user login feature.
- Use the logged-in user ID to obtain the list of contacts from the database.
- Provide a UI option to create a new contact and save it for future use.
- Extend the app with voice features by using the voice APIs of external services.

If you have other ideas for improvements, please add them in the article comments section. Even better, join my multimessage project and update the code!

The SQL Database service [http://www.ibm.com/developerworks/topics/sql database service](http://www.ibm.com/developerworks/topics/sql database service) adds an on-demand relational database to your application. Powered by DB2, it provides a managed database service to handle demanding web and transactional workloads.

**RELATED TOPICS:** WASdev developer center  Bluemix messaging
About the author

Ahmad Abdul Wakeel

&nbsp;

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)