

# Build a DB2 for z/OS mobile application using IBM MobileFirst

Jane Man ([janeman@us.ibm.com](mailto:janeman@us.ibm.com))

Senior Software Engineer  
IBM

21 September 2015

Clement Leung ([clement.leung96@gmail.com](mailto:clement.leung96@gmail.com))

Author  
Consultant

A new era of applications focused on mobile devices is becoming increasingly important in large enterprises. Learn how you can build a DB2® for z/OS® mobile application using IBM MobileFirst Developer Edition. Using an easy-to-adapt example, this tutorial shows what needs to be done from the front end to the back-end database server to make this happen.

## Introduction

Mobile is a pervasive tool that most people keep within reach most of the time, even while sleeping. We use our phones to browse from social media sites to shopping sites to gather information for making decisions. Mobile has changed the business landscape, causing CIOs to rethink how they conduct business. Mobile workloads can increase transaction rates from 10-50 percent — particularly read-only transactions by users constantly checking to see if a balance has been updated or whether a package is on its way. There may be unanticipated spikes in workload due to some special offers.

DB2 for z/OS is commonly used to store critical transaction data. A large number of data transactions in banking, retail, insurance, and government, etc. occur on DB2 for z/OS because of its quality of service, high availability, security, and performance.

The IBM MobileFirst Platform provides an integrated development and testing environment for mobile applications. The development environment is simplified by combining multiple set of tools, frameworks, and code bases into a single development environment, and only one code base to develop and maintain. It provides an Eclipse-based studio and command-line interface for:

- Native application development:
  - iOS
  - Android

- Windows Phone 8
- Windows 8
- Hybrid development (A combination of native apps and web app development)
- Server-side development (adapters):
  - SQL Adapter
  - HTTP adapters
  - Java™ adapters
- Mobile Console Browser (for testing and debugging)

In the following sections, we illustrate how to build a DB2 for z/OS mobile application using IBM MobileFirst Developer Edition for the following two simple scenarios, which can be used as a basis for building more complex applications:

1. Retrieve string ID from SYSIBM.SYSXMLSTRINGS catalog table using regular SQL statements.
2. Execute `RUNSTATS` for a particular database and tablespace by calling a stored procedure.

We will cover the following topics:

- Installation and configuration
- Client-side development
- Server-side development
- Using server-side result
- Building, deploying, testing, and using IBM MobileFirst console
- Running an Android emulator (optional)

## Installation and configuration

### Installation

IBM MobileFirst Developer Edition provides two development tools: Eclipse-based IDE (Developer Studio) and command-line interface (CLI), which is intended for more advanced users. In this tutorial, we focus on IBM MobileFirst Developer Studio.

To install IBM MobileFirst 7 Developer Studio:

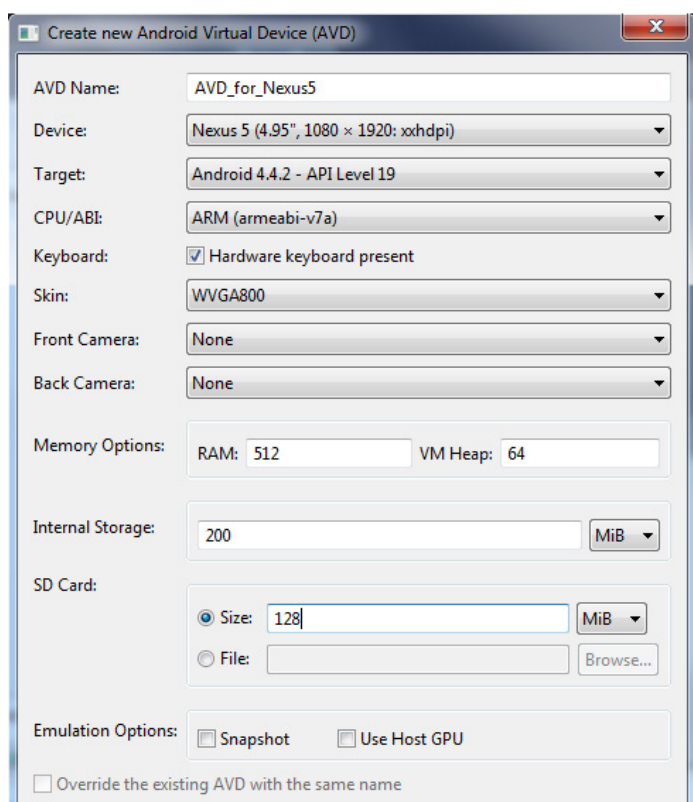
1. Install JRE 7
  - Oracle JRE 7 is required for developing an Android native app.
  - Make sure you install JRE7 and not JRE 8. The server does not start with the JRE 8. (at the time of this writing).
2. Install Eclipse IDE for Java EE Developers (one of the following): Juno SR2 (4.2.2), Kepler SR1 (4.3.1), Kepler SR2 (4.3.2), Luna SR1 (4.4.1), or Luna SR2 (4.4.2) Third Topic  
For this tutorial, we installed Juno SR2 (4.2.2).
3. Start Eclipse, then select **Help > Eclipse Marketplace**.
  - a. In the **Find** field, type `MobileFirst Platform` and click **Go**.
  - b. (Optionally install: IBM Dojo Mobile Tools and IBM jQuery Mobile Tools.
4. (Optionally install and configure Android native development: Android SDK, ADT Eclipse plug-in, creating Android Virtual Device (AVD). See details below.

To connect to DB2 for z/OS, you need db2jcc.jar and db2jcc\_license\_cisuz.jar (from DB2 Connect). Get more information about [MobileFirst installation](#).

Optional: If you decide to set up the Android development environment (with Oracle JRE 7):

1. Install the [Android SDK](#). Scroll to **Other Download Option > SDK Tools only**.
2. Install the ADT Eclipse plug-in and Inside Eclipse:
  - Click **Help > Install New Software**.
  - Click **Add**, enter location: <https://dl-ssl.google.com/android/eclipse/>.
  - Select **Developer Tools** and follow the installation prompt.
3. Add the SDK. Inside Eclipse, click **Window > Android SDK Manager**. By default, it installs the latest API level, but I also install API level 19.
4. Add a virtual device (emulator). Inside Eclipse, click **Window > Android Virtual Device Manager**. Click **Create** button to create a new AVD. The below figure is an example of an AVD I used.

**Figure 1. Example of an Android Virtual Device (AVD)**



## Configure JNDI for DB2 for z/OS

Edit `MobileFirst Development Server > servers > worklight > server.xml` to add `<library>`, `<dataSource>`, similar to the code listing below. We will reference the database defined here later.

```

<library id="db2jcc">
<fileset dir="C:\JCCJars\JCC411" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>

<dataSource id="db2" jndiName="dtec222">
  <jdbcDriver libraryRef="db2jcc"/>
  <properties.db2.jcc databaseName="STLEC1" password="password1" portNumber="446"
serverName="hostname1.svl.ibm.com" user="sysadm"/>
</dataSource>

<dataSource id="db2_serveros" jndiName="zserveros">
  <jdbcDriver libraryRef="db2jcc"/>
  <properties.db2.jcc databaseName="E0SDBV11" password="password2" portNumber="5456"
serverName="hostname2.ibm.com" user="I0D02S"/>
</dataSource>

```

## Scenario 1

### Scenario 1: Retrieve string from string ID from the SYSIBM.SYSXMLSTRINGS table using regular SQL statements

SYSIBM.SYSXMLSTRINGS is a catalog table that contain mapping between string and string ID used inside XML storage. There is a set of pre-defined mapping when DB2 is installed. The below image is an output example of the SYSIBM.SYSXMLSTRINGS catalog table.

**Figure 2. Output of the SYSIBM.SYSXMLSTRINGS table**

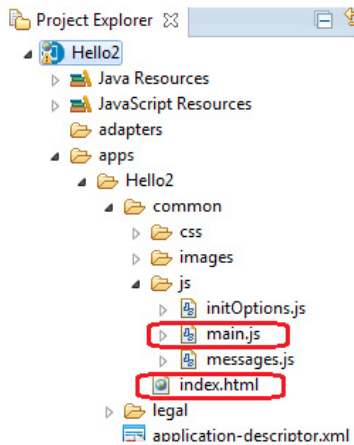
File Edit Format View Help		
SELECT STRINGID, SUBSTR(STRING,1,60)as STRING, IBMREQD		
FROM SYSIBM.SYSXMLSTRINGS		
STRINGID	STRING	IBMREQD
1001	product	N
1002	description	N
1003	name	N
1004	detail	N
1005	a	N
1006	instruction	N
1007	step	N
1008	http://www.w3.org/2000/xmlns/	N
1009	space	N
1010	purchaseOrder	N
1011	orderDate	N
1012	shipTo	N
1013	country	N
1014	street	N
1015	city	N

## Client-side development

As mentioned, we focus on using IBM MobileFirst Studio for our development. Create an IBM MobileFirst project:

1. Click **File > New > MobileFirst Project**.
2. Enter Hello2 as the project name.
3. Select **Hybrid Application**.
4. Enter Hello2 as the application name.

### Figure 3. Files layout for a newly created project



The above image illustrates the files layout for a newly created project. The front-end display is defined in `index.html` under `apps > Hello2 > common`.

Inside `index.html`, as shown below, we define a text field to enter a string ID value.

```
<body style="display: none;">
  <div data-role="page" id="page">
    <div data-role="content" style="padding: 15px">
      
      <br/>
      Find DB2 for z/OS XML String from String id
    </div>

    <label for="text">String id:</label>
    <input type="text" name="text" id="stringid"/>

    <input type="button" value="Submit" src="js/main.js"
    onclick="loadFeeds(document.getElementById('stringid').value)"/>
    <div id="wrapper">
      <ul id="itemsList"/>
    </div>
  </div>

  <script src="js/initOptions.js"/>
  <script src="js/main.js"/>
  <script src="js/messages.js"/>
</body>
```

When the **Submit** button is clicked, the `loadFeeds` function in `main.js` is called as shown below.

```
function loadFeeds(stringid){
  WL.Logger.debug("Inside loadFeeds");
  busyIndicator.show();
  stringidInput = stringid;

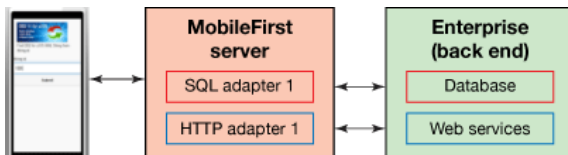
  var invocationData = {
    adapter : 'Hello2DB2Adapter',
    procedure : 'getStringFromStringId',
    parameters : [stringid]
  };

  WL.Client.invokeProcedure(invocationData, {
    onSuccess : loadFeedsSuccess,
    onFailure : loadFeedsFailure
  });
}
```

As shown, inside `loadFeeds`, the procedure `getStringFromStringId` is called within `Hello2DB2Adapter`. An adapter is a server-side technology we will discuss in the next section.

## Server-side development

**Figure 4. IBM MobileFirst architecture**



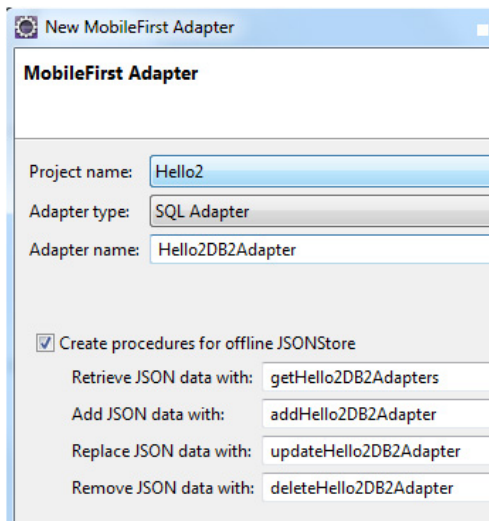
The above image is a simplified architecture of IBM MobileFirst. A mobile application needs to access a server that provides services it needs. IBM MobileFirst provides a set of adapters for server-side development. In this scenario, we need to create a SQL adapter to connect to DB2 for z/OS.

## Create an SQL adapter

To create an SQL adapter, inside Eclipse:

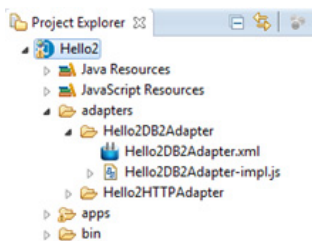
1. Click **File > New > MobileFirst Adapter**.
2. Select `Hello2` as project name.
3. Select **SQL Adapter**.
4. Enter `Hello2DB2Adapter` as adapter name.
5. Check **Create procedure for offline JSONStore**.

## Figure 5. Creating an SQL adapter



After an SQL adapter is created, there will be two files under the newly created adapter: Hello2DB2Adapter.xml and Hello2Adapter-impl.js. as illustrated below.

## Figure 6. File layout of creating a new adapter



1. Edit Hello2DB2Adapter.xml to:

- Add zserveros as dataSourceJNDIName. This should be the same as the one defined by the jndiName attribute in <dataSource> in server.xml.
- Add a procedure call getStringFromStdString.

```
<connectivity>
  <connectionPolicy xsi:type="sql:SQLConnectionPolicy">
    <dataSourceJNDIName>zserveros</dataSourceJNDIName>
  </connectionPolicy>
</connectivity>

<procedure name="procedure1"/>
<procedure name="procedure2"/>
<procedure name="getHello2DB2Adapters"> </procedure>
<procedure name="addHello2DB2Adapter"> </procedure>
<procedure name="updateHello2DB2Adapter"> </procedure>
<procedure name="deleteHello2DB2Adapter"> </procedure>
<procedure name="getStringFromStdString"></procedure>
```

1. Edit Hello2DB2Adapter-impl.js to implement getStringFromStdString, as below.

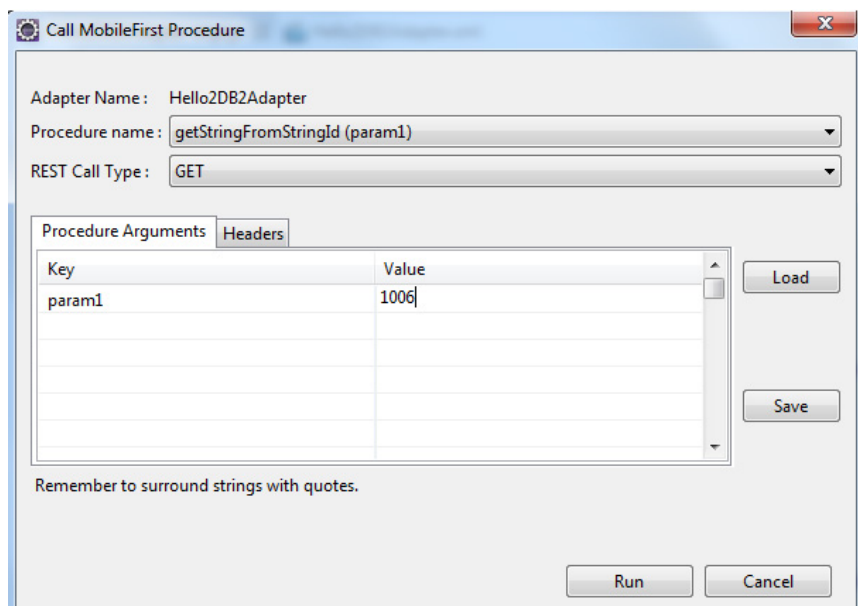
```
var select2Statement = WL.Server.createSQLStatement(  
    "SELECT SUBSTR(STRING,1,60) as STRING from SYSIBM.SYSXMLSTRINGS WHERE STRINGID= ?");  
  
function getStringFromStringId(param1)  
{  
    return WL.Server.invokeSQLStatement  
    ({  
        preparedStatement : select2Statement,  
        parameters : [param1]  
    });  
}
```

## Deploy and test an adapter

1. Right-click the **Hello2DB2Adapter > Run As > Deploy MobileFirst Adapter**.
2. Right-click **Hello2DB2Adapter > Run As > Call MobileFirst Adapter**.

Enter the value of the parameter, then click **Run**. A GUI like below would appear.

### Figure 7. Call IBM MobileFirst procedure GUI



The result, as shown below, appears in the JSON format.

```
{  
  "isSuccessful": true,  
  "resultSet":  
  [  
    {  
      "STRING": "instruction"  
    }  
  ]  
}
```

## Use the result from the server side

As shown, when `invokeProcedure` throws success, `loadFeedsSuccess` is called.



In order to utilize the result after calling an adapter or external resource, we need to edit `main.js`, implement `loadFeedsSuccess`, as below. Note the `resultSet` from a SQL adapter is a `result.invocationResult.resultSet` object. We first check the length of this object. If it is not empty, we call `displayFeeds()` with `resultSet` object as parameter.

The following listing shows the implementation of `loadFeedsSuccess` in `main.js`.

Inside `displayFeeds()`, we call `WL.SimpleDialog.show()` to display the first value.

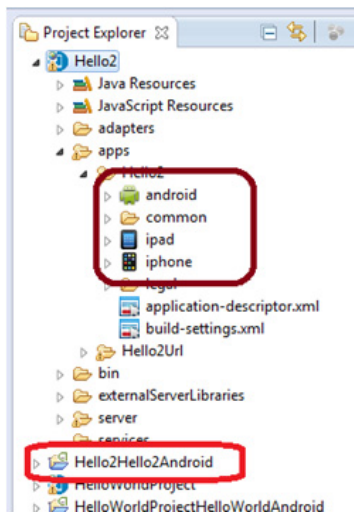
```
function loadFeedsSuccess(result){
    WL.Logger.debug("Feed retrieve success");
    busyIndicator.hide();
    if (result.invocationResult.resultSet.length>0)
        displayFeeds(result.invocationResult.resultSet);
    else if (result.invocationResult.resultSet.length==0)
        loadFeedsNotFound();
    else
        loadFeedsFailure();
}
function displayFeeds(resultSet){
    var ul = $('#itemsList');

    WL.SimpleDialog.show(resultSet[0].STRING, "String id for " + stringidInput + " is found.",
        [{
            text : 'Reload',
            handler : WL.Client.reloadApp
        }
    ]);
}
```

## Build and deploy

1. Right-click **Hello2\apps\Hello2 > Run As > Build All Environments**.
2. Select **iPhone, iPad, Android phones, and tablets**.

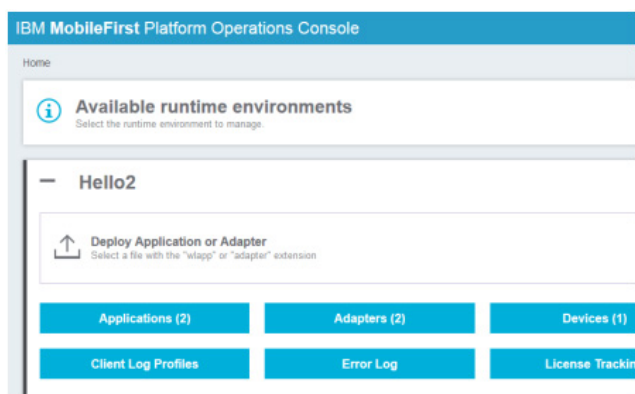
After a successful build, you should see folders for Android, iPad, and iPhone, as shown below. A project called `Hello2Hello2Android` is created as well. (Note: The first `Hello2` is the original project name and the second is the original application name.)

**Figure 8. Folders after building all the environments**

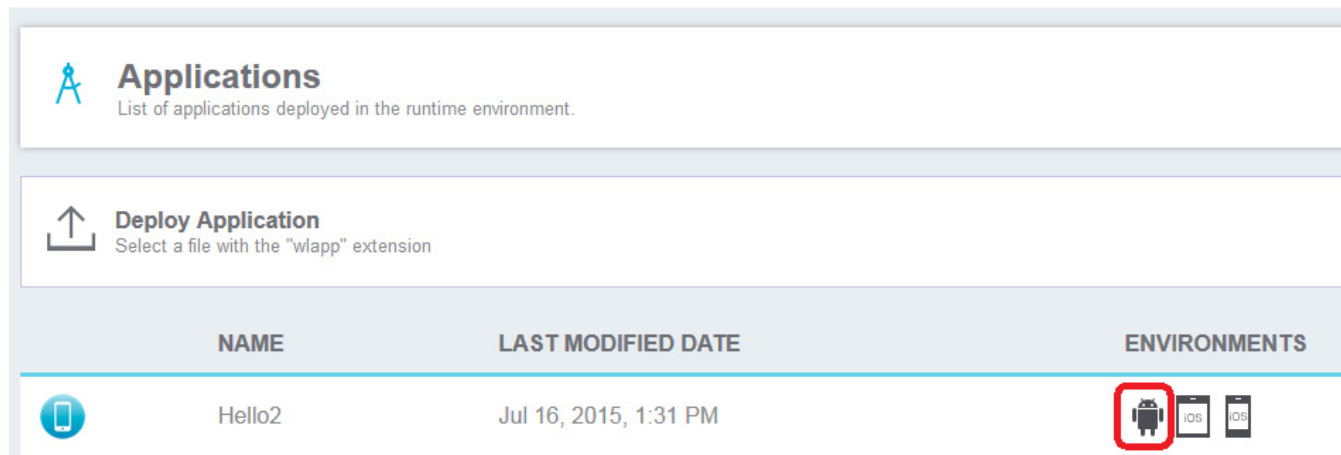
## Test using IBM MobileFirst console

1. Right-click **Hello2** > **Open MobileFirst Console**.




A browser similar to the one below will appear.

**Figure 9. IBM MobileFirst console**

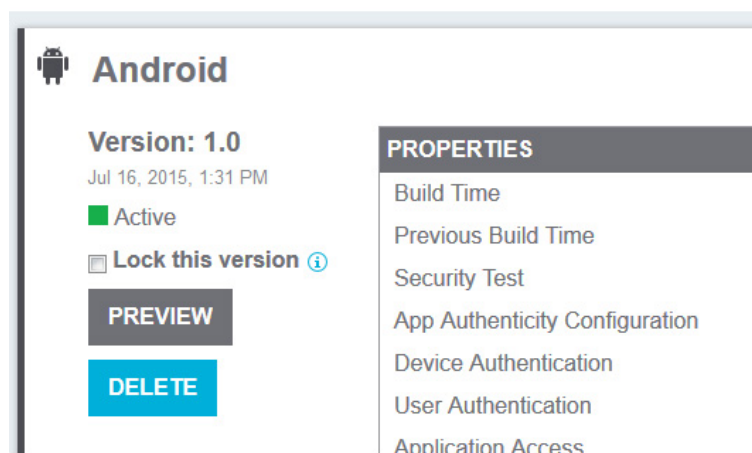
1. Click **Applications**.
2. Click **Android** for Hello2, as shown below.

**Figure 10. Console for Hello2 application**

The screenshot shows the 'Applications' section of the IBM MobileFirst console. At the top, there's a 'Deploy Application' button with an upload icon and the text 'Select a file with the "wapp" extension'. Below this is a table with columns: NAME, LAST MODIFIED DATE, and ENVIRONMENTS. The table contains one entry for 'Hello2', which was last modified on 'Jul 16, 2015, 1:31 PM'. In the 'ENVIRONMENTS' column, there are three icons: an Android icon (highlighted with a red box), an iOS icon, and another iOS icon.

NAME	LAST MODIFIED DATE	ENVIRONMENTS
Hello2	Jul 16, 2015, 1:31 PM	  

1. Click the **Preview** button.

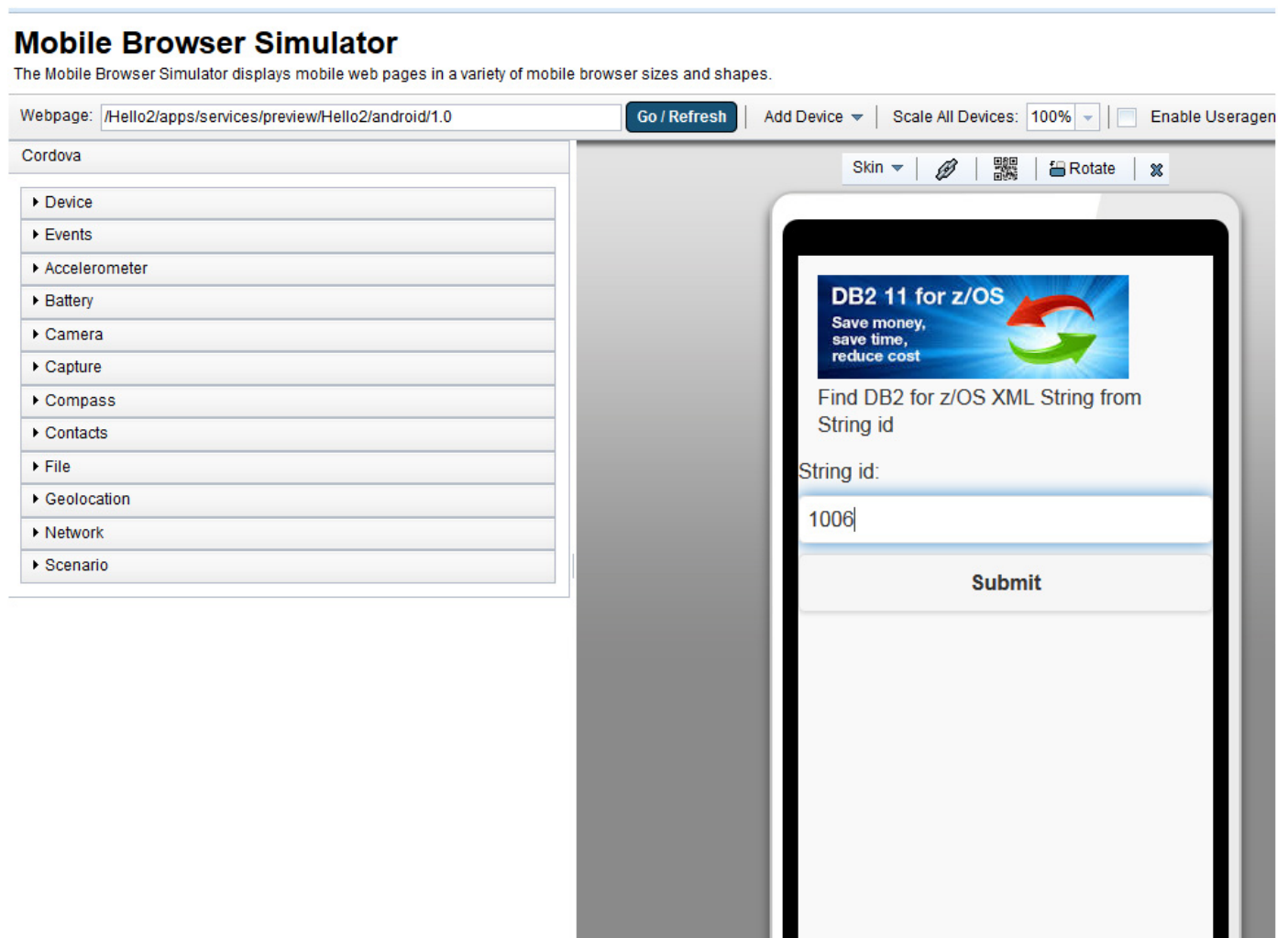
**Figure 11. Android Simulator under Hello2 application**

The screenshot shows the 'Android' simulator settings for the 'Hello2' application. On the left, there's a sidebar with the 'Android' icon and title. Below it, the 'Version: 1.0' is displayed, along with the date 'Jul 16, 2015, 1:31 PM'. There's a green 'Active' status indicator and a 'Lock this version' checkbox. At the bottom of the sidebar are 'PREVIEW' and 'DELETE' buttons. On the right, there's a 'PROPERTIES' section with a list of settings: Build Time, Previous Build Time, Security Test, App Authenticity Configuration, Device Authentication, User Authentication, and Application Access.

**Android**  
Version: 1.0  
Jul 16, 2015, 1:31 PM  
Active  
☐ Lock this version ⓘ  
**PREVIEW**  
**DELETE**

**PROPERTIES**  
Build Time  
Previous Build Time  
Security Test  
App Authenticity Configuration  
Device Authentication  
User Authentication  
Application Access

A mobile browser simulator like below will appear, and we can test the new Hello2 application.

**Figure 12. Android simulator**

## Run Android emulator (optional)

If the Android SDK ADT Eclipse plug-in has been installed and an Android virtual device is defined, you can run the Android emulator as well.

1. Right-click **Hello2Hello2Android > Run As > Android Application**.

The following image illustrates the screen you will see on the Android emulator. Depending on the CPU of your machine, response time may vary.

**Figure 13. Android emulator**

## Scenario 2

### Scenario 2: Execute runstats for a particular database and table space by calling a stored procedure

1. Follow the instructions above to create a new IBM MobileFirst project and a new hybrid application called Runstats.
2. Create a SQL adapter for Runstats project called RunstatsSQLAdapter.
3. Edit RunstatsSQLAdapter.xml to:
  - Add `zserveros` as `dataSourceJNDIName` (or whatever JNDI name you defined in `server.xml`).
  - Add `<procedure name="runstatSP1"/>`.
4. Edit `RunstatsSQLAdapter-impl.js` to implement `runstatSP1`. We need to call a procedure called `SYSPROC.DSNUTILS` using `WL.Server.invokeSQLStoredProcedure()` and pass in the parameters. The listing below shows part of its implementation.  
In addition to `runstats`, other utilities can be called in the same way.

The following listing shows the implementation of `runstatSP1`.

```
function runstatSP1(database, tablespace) {
    var utilityStatement = "RUNSTATS TABLESPACE " + database + "." + tablespace + " UPDATE(ALL)";
    utilityStatement = utilityStatement.toString();

    return WL.Server.invokeSQLStoredProcedure({
        procedure : "SYSPROC.DSNUTILS",
        parameters : [123456789,
            "",
            utilityStatement,
            "",
            "RUNSTATS TABLESPACE",
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            "",
            0,
            "",
            ""],
    });
}
```

```

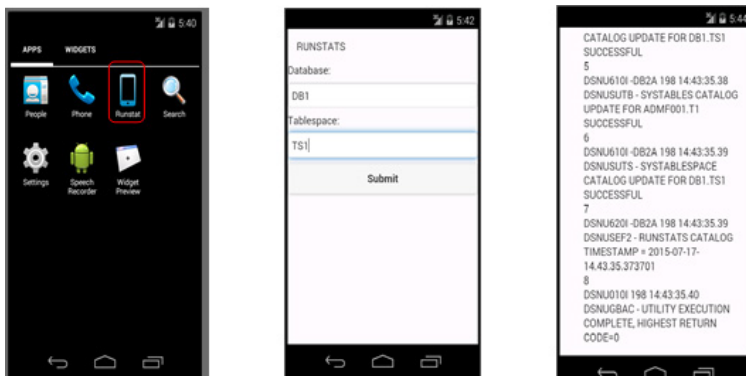
        "" ,
        0 ,
        "" ,
        "" ,
        0 ,
    ]
});
}

```

5. Implement the front end (client side) similar to scenario 1. In index.xml, we need two inputs: one for the database and one for the table space. Main.js needs to be modified to utilize the result after invoking `runstatSP1`.

Figure 14 illustrates the screen you will use when you run this application on an Android emulator.

**Figure 14. Execute `runstats` in Android emulator**



## Summary

This tutorial discusses how to build a DB2 for z/OS mobile application and the related server-side development using IBM MobileFirst 7 Developer Edition. We illustrated how to install, configure, develop, build, and test the mobile application.

## Acknowledgments

Thanks to Susan Malaika for her comments and assistance with this tutorial.

## Resources

- Check out [IBM MobileFirst 7 Developer Edition](#).
- Learn more about [Native Android Development](#).
- Check out the [JavaScript SQL Adapter](#).
- Visit the Knowledge Center to learn more about [SYSPROC.DSNUTILS](#).
- The [Information Management area on developerWorks](#) provides resources for architects, developers, and engineers.
- Stay current with [developer technical events and webcasts](#) focused on a variety of IBM products and IT industry topics.
- Follow [developerWorks on Twitter](#).
- Watch [developerWorks demos](#) ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Get involved in the [developerWorks Community](#). Connect with other developerWorks users while you explore developer-driven blogs, forums, groups, and wikis.

## About the authors

### Jane Man

Jane Man is a senior software engineer in development team in DB2 for z/OS. She has worked on various features of DB2 for z/OS. In addition to her development work, she is the enablement focal point and is involved in many enablement activities, including creating sample applications, demos, Hands on Labs, and presenting in conferences and bootcamps, etc. Before joining DB2 for z/OS, she was a developer in IBM Content Manager. Jane is a IBM Certified System Administrator for WebSphere Application Server; IBM Certified Database Administrator for DB2 Universal Database for z/OS, Linux, UNIX and Windows; IBM Certified Solution Designer for DB2 Content Manager; IBM Certified Deployment Profession for Tivoli Storage Manager; and IBM Certified Application developer for DB2 Universal Database Family.

---

### Clement Leung

Clement Leung is a Mission San Jose High School graduate and is planning to study computer science this fall.

© Copyright IBM Corporation 2015

([www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml))

Trademarks

([www.ibm.com/developerworks/ibm/trademarks/](http://www.ibm.com/developerworks/ibm/trademarks/))