# JavaScript SQL Adapter

#### **Overview**

An IBM MobileFirst Platform Foundation SQL adapter is designed to communicate with any SQL data source. You can use plain SQL queries or stored procedures.

To connect to a database, JavaScript code needs a JDBC connector driver for the specific database type. You must download the JDBC connector driver for the specific database type separately and add it as a dependency in your project. For more information on how to add a dependency, see the Dependencies section in the Creating Java and JavaScript Adapters (../../#dependencies) tutorial.

In this tutorial and in the accompanying sample, you learn how to use a MobileFirst adapter to connect to a MySQL database.

**Prerequisite:** Make sure to read the JavaScript Adapters (../) tutorial first.

#### The XML File

The XML file contains settings and metadata.

- 1. In the adapter XML file, declare the following parameters:
  - Driver Class
  - Database URL
  - Username
  - Password

```
<?xml version="1.0" encoding="UTF-8"?>
<mfp:adapter name="JavaScriptSQL"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:mfp="http://www.ibm.com/mfp/integration"
  xmlns:sql="http://www.ibm.com/mfp/integration/sql">
  <displayName>JavaScriptSQL</displayName>
  <description>JavaScriptSQL</description>
  <connectivity>
    <connectionPolicy xsi:type="sql:SQLConnectionPolicy">
      <dataSourceDefinition>
         <driverClass>com.mysql.jdbc.Driver</driverClass>
         <url>jdbc:mysql://localhost:3306/mobilefirst training</url>
         <user>mobilefirst</user>
         <password>mobilefirst</password>
      </dataSourceDefinition>
    </connectionPolicy>
  </connectivity>
  cprocedure name="getAccountTransactions1"/>
  countTransactions2"/>
</mfp:adapter>
```

2. Declare a procedure in the adapter XML file.

```
cprocedure name="getAccountTransactions1"/>
```

### JavaScript implementation

The adapter JavaScript file is used to implement the procedure logic.

There are two ways of running SQL statements:

- SQL statement query
- SQL stored procedure
- 1. Use the WL.Server.createSQLStatement method to prepare a SQL query. This method must always be called outside the function.
- 2. Add more parameters, if necessary.

```
//Create SQL query
var getAccountsTransactionsStatement = WL.Server.createSQLStatement(
"SELECT transactionId, fromAccount, toAccount, transactionDate, transactionAmount, transactionT
ype " +
"FROM accounttransactions " +
"WHERE accounttransactions.fromAccount = ? OR accounttransactions.toAccount = ? " +
"ORDER BY transactionDate DESC " +
"LIMIT 20;"
);
```

- 3. Use the WL.Server.invokeSQLStatement method to call prepared queries.
- 4. Return the result to the application or to another procedure.

```
//Invoke prepared SQL query and return invocation result

function getAccountTransactions1(accountId){
    return WL.Server.invokeSQLStatement({
        preparedStatement : getAccountsTransactionsStatement,
        parameters : [accountId, accountId]
    });
}
```

- 5. To run a SQL stored procedure, use the WL.Server.invokeSQLStoredProcedure method. Specify a SQL stored procedure name as an invocation parameter.
- 6. Add more parameters, if necessary.
- 7. Return the invocation result to the application or to another procedure.

```
//Invoke stored SQL procedure and return invocation result
function getAccountTransactions2(accountId){
  return WL.Server.invokeSQLStoredProcedure({
    procedure : "getAccountTransactions",
    parameters : [accountId]

});
}
```

### **Invocation Results**

The result is retrieved as a JSON object:

```
"isSuccessful": true,
 "resultSet": [{
  "fromAccount": "12345",
  "toAccount": "54321",
  "transactionAmount": 180.00,
  "transactionDate": "2009-03-11T11:08:39.000Z",
  "transactionId": "W06091500863",
  "transactionType": "Funds Transfer"
  "fromAccount": "12345",
  "toAccount": null,
  "transactionAmount": 130.00,
  "transactionDate": "2009-03-07T11:09:39.000Z",
  "transactionId": "W214122\/5337",
  "transactionType": "ATM Withdrawal"
 }]
}
```

- The isSuccessful property defines whether the invocation was successful.
- The resultSet object is an array of returned records.
  - To access the resultSet object on the client-side: result.invocationResult.resultSet
  - To access the resultSet object on the server-side: result.ResultSet

## Sample application

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/JavaScriptAdapters) the MobileFirst project.

- The Adapters project also includes a sample MySQL script in the **Utils** folder, which needs to be imported into your database to test the project.
- Make sure that the mobilefirst@% user has all access permissions assigned to it.
- Remember to download and set the MySQL Java Connector in your Adapters project.