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.
As a developer, you must download the JDBC connector driver for the specific database type separately and add it to the `server\lib\` folder of a MobileFirst project.
You can download the JDBC connector driver from the appropriate vendor website.
In this tutorial and in the accompanying sample, you learn how to use a MobileFirst adapter to connect to a MySQL database.

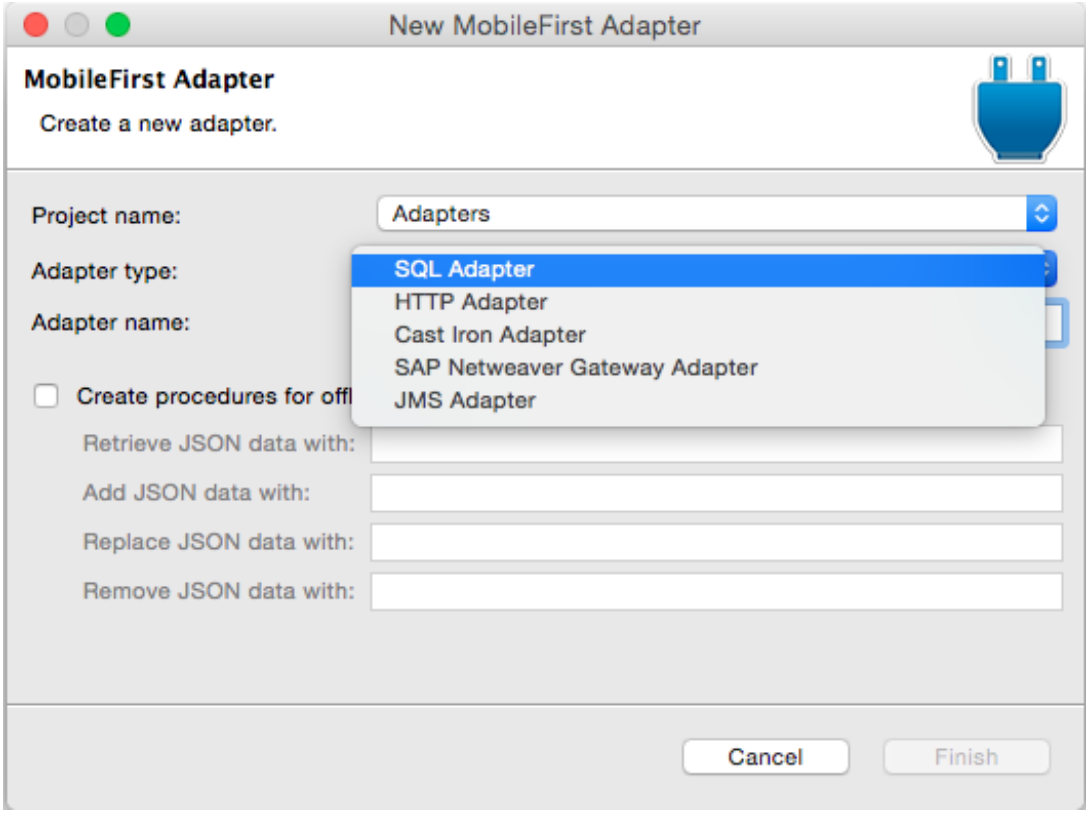
- Creating the adapter
- Adapter XML
- JavaScript implementation file
- Invocation results
- Sample application

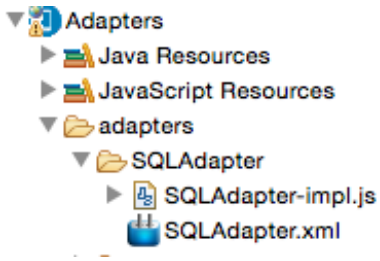
Creating the adapter

CLI

From the project's directory, use `mfp add adapter` and follow the interactive instructions.

Studio

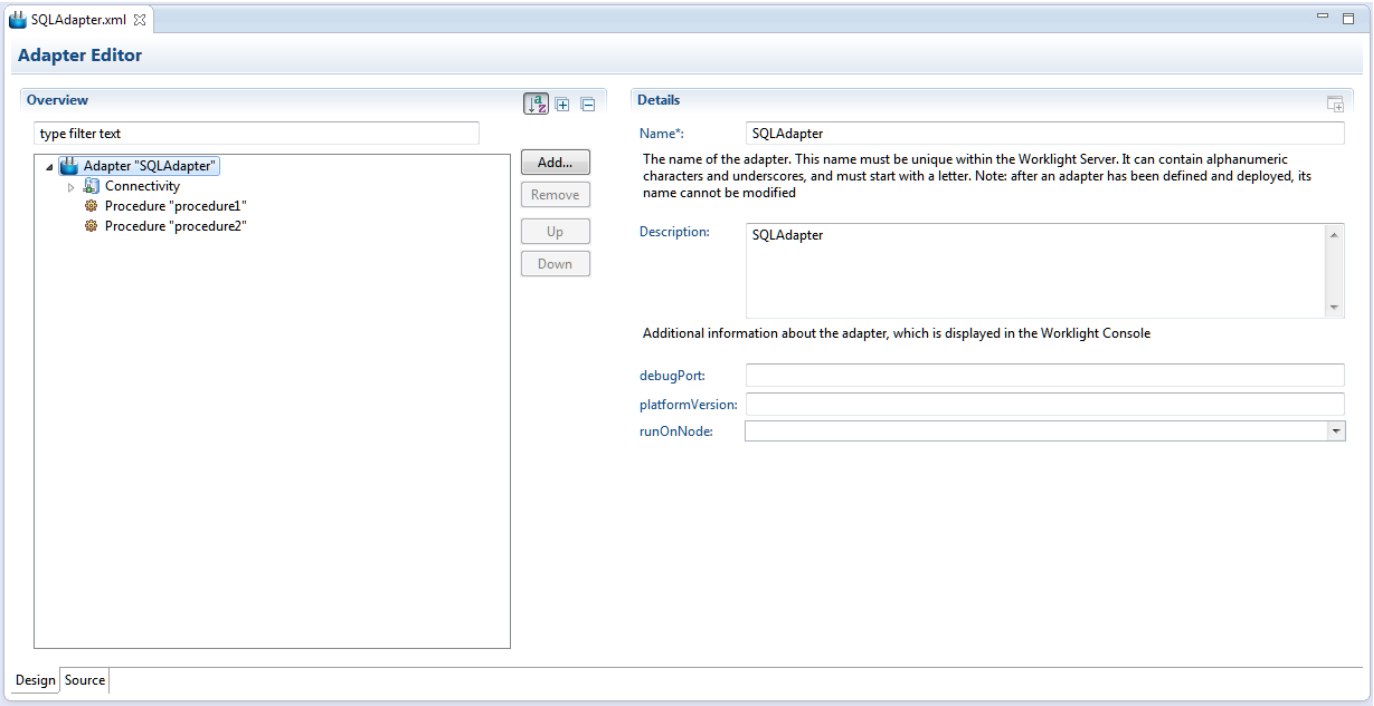




1. In MobileFirst Studio, create a MobileFirst adapter and choose the SQL adapter type.
   A standard SQL adapter file structure is created.
2. Save the downloaded JDBC connector file in the project `server\lib\` folder.

Adapter XML

Settings and metadata are stored in the adapter XML file.
You can use either the Design or the Source editor in the Studio, or use your favorite XML editor to modify the adapter XML file.



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

```
<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>
```

2. Declare a procedure in the adapter XML file.

```
<procedure name="getAccountTransactions1"/>
```

JavaScript implementation file

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

Important: The name that is declared in the XML file must be used for the procedure JavaScript function.

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.
   The `WL.Server.createSQLStatement` 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, transactio
nType " +
  "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


```
{
  "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": "W214122V5337",
    "transactionType": "ATM Withdrawal"
  }]
}
```


The result is retrieved as a JSON object.

- 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/tree/release71) the MobileFirst project.

The sample project contains an SQL adapter.

- To run the sample, execute the `mobilefirstTraining.sql` file (which you can find under the Server folder of the sample) on your local MySQL server.
- Make sure that the `mobilefirst@% user` has all access permissions assigned to it.
- Remember to download and set the MySQL Java Connector in your project.