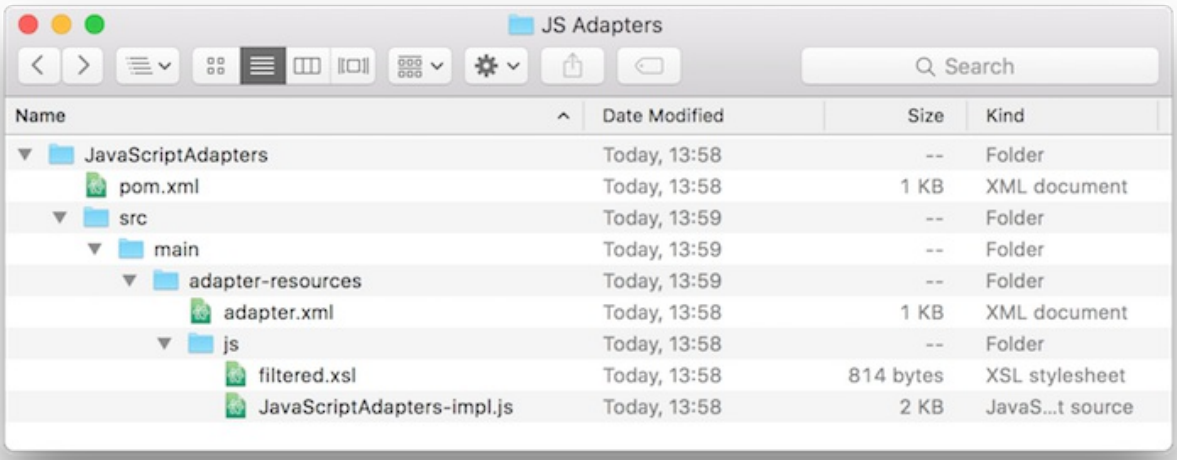
JavaScript Adapters

Overview

JavaScript adapters provide templates for connection to HTTP and SQL back-ends. It provides a set of services, called procedures and mobile apps can call these procedures by issuing AJAX requests.

Prerequisite: Make sure to read the Creating Java and JavaScript Adapters (../creating-adapters) tutorial first.

File structure



The adapter-resources folder

The `adapter-resources` folder contains an XML configuration file. This configuration file describes the connectivity options and lists the procedures that are exposed to the application or other adapters.

```
<?xml version="1.0" encoding="UTF-8"?>
<mfp:adapter name="JavaScriptAdapter">
    <displayName>JavaScriptAdapter</displayName>
    <description>JavaScriptAdapter</description>

    <connectivity>
        <connectionPolicy>
        ...
        </connectionPolicy>
    </connectivity>

    <procedure name="procedure1"></procedure>
    <procedure name="procedure2"></procedure>

    <property name="name" displayName="username" defaultValue="John"  />
</mfp:adapter>
```

Click for adapter.xml attributes and subelements

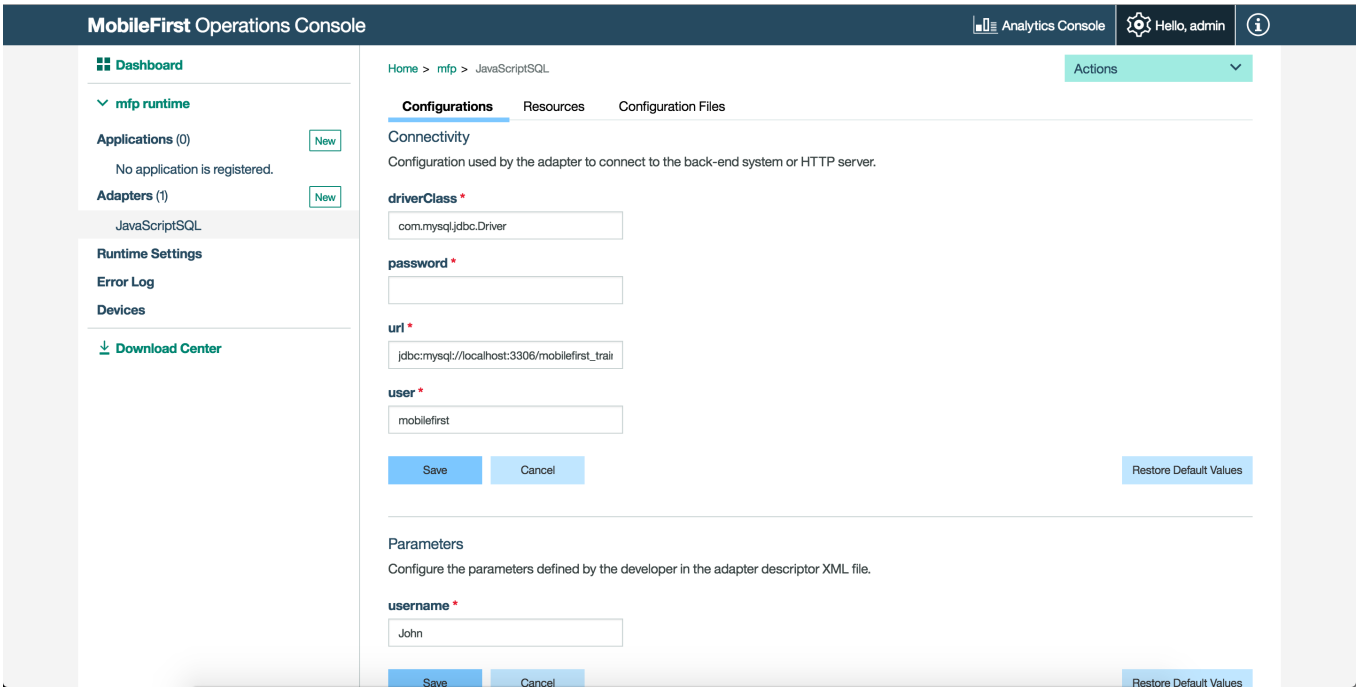
Custom properties

The **adapter.xml** file can also contain user-defined custom properties. The values that developers assign to them during the creation of the adapter can be overridden in the **MobileFirst Operations Console → [your adapter] → Configurations tab**, without redeploying the adapter. User-defined properties can be read using the getPropertyValue API and then further customized at run time.

> **Note:** The configuration properties elements must always be located *below* the `procedure` elements. In the example above we defined a displayName property with a default value, so it could be used later.

The `<property>` element takes the following attributes:

- **name**: The name of the property, as defined in the configuration class.
- **defaultValue**: Overrides the default value defined in the configuration class.
- **displayName**: *optional*, a friendly name to be displayed in the console.
- **description**: *optional*, a description to be displayed in the console.
- **type**: *optional*, ensures that the property is of a specific type such as `integer`, `string`, `boolean` or a list of valid values (for example `type="['1','2','3']"`).



Pull and Push Configurations

Customized adapter properties can be shared using the adapter configuration file found in the **Configuration files tab**.
To do so, use the `pull` and `push` commands described below using either Maven or the MobileFirst CLI.
For the properties to be shared, you need to *change the default values given to the properties*.

Run the commands from the root folder of the adapter Maven project:

Maven

- To **pull** the configurations file - `mvn adapter:configpull -DmfpfConfigFile=config.json`.
- To **push** the configurations file - `mvn adapter:configpush -DmfpfConfigFile=config.json`.

MobileFirst CLI

- To **pull** the configurations file - `mfpdev adapter pull`.
- To **push** the configurations file - `mfpdev adapter push`.

> Learn more in by using `mfpdev help adapter pull/push`.

The js folder

This folder contains all the JavaScript implementation file of the procedures that are declared in the **adapter.xml** file. It also contains zero, one, or more XSL files, which contain a transformation scheme for retrieved raw XML data. Data that is retrieved by an adapter can be returned raw or preprocessed by the adapter itself. In either case, it is presented to the application as a **JSON object**.

JavaScript adapter procedures

Procedures are declared in XML and are implemented with server-side JavaScript, for the following purposes:

- To provide adapter functions to the application
- To call back-end services to retrieve data or to perform actions

Each procedure that is declared in the **adapter.xml** file must have a corresponding function in the JavaScript file.

By using server-side JavaScript, a procedure can process the data before or after it calls the service. You can apply more filtering to retrieved data by using simple XSLT code.
JavaScript adapter procedures are implemented in JavaScript. However, because an adapter is a server-side entity, it is possible to use Java in the adapter (../javascript-adapters/using-java-in-javascript-adapters) code.

Using global variables

The MobileFirst server does not rely on HTTP sessions and each request may reach a different node. You should not rely on global variables to keep data from one request to the next.

Adapter response threshold

Adapter calls are not designed to return huge chunks of data because the adapter response is stored in MobileFirst Server memory as a string. Thus, data that exceeds the amount of available memory might cause an out-of-memory exception and the failure of the adapter invocation. To prevent such failure, you configure a threshold value from which the MobileFirst Server returns gzipped HTTP responses. The HTTP protocol has standard headers to support gzip compression. The client application must also be able to support gzip content in HTTP.

Server-side

In the MobileFirst Operations Console, under **Runtimes > Settings > GZIP compression threshold for adapter responses**, set the desired threshold value. The default value is 20 KB.
Note: By saving the change in the MobileFirst Operations Console, the change is effective immediately in the runtime.

Client-side

Ensure that you enable the client to parse a gzip response, by setting the value of the `Accept-Encoding` header to `gzip` in every client request. Use the `addHeader` method with your request variable, for example: `request.addHeader("Accept-Encoding","gzip");`

Server-side APIs

JavaScript adapters can use the MobileFirst server-side APIs to perform operations that are related to MobileFirst Server, such as calling other JavaScript adapters, logging to the server log, getting values of configuration properties, reporting activities to Analytics and getting the identity of the request issuer.

getPropertyValue

Use the `MFP.Server.getPropertyValue(propertyName)` API to retrieve properties defined in the **adapter.xml** or in the MobileFirst Operations Console:

```
MFP.Server.getPropertyValue("name");
```

getAdapterName

Use the `getAdapterName()` API to retrieve the adapter name.

invokeHttp

Use the `MFP.Server.invokeHttp(options)` API in HTTP adapters.
You can see usage examples on the JavaScript HTTP Adapter (js-http-adapter) tutorial.

invokeSQL

Use the `MFP.Server.invokeSQLStatement(options)` and the `MFP.Server.invokeSQLStoredProcedure(options)` APIs in SQL adapters.
You can see usage examples on the JavaScript SQL Adapter (js-sql-adapter) tutorial.

addResponseHeader

Use the `MFP.Server.addResponseHeader(name,value)` API to add a new header(s) to the response:

```
MFP.Server.addResponseHeader("Expires","Sun, 5 October 2014 18:00:00 GMT");
```

getClientRequest

Use the `MFP.Server.getClientRequest()` API to get a reference to the Java HttpServletRequest object that was used to invoke an adapter procedure:

```
var request = MFP.Server.getClientRequest();
var userAgent = request.getHeader("User-Agent");
```

invokeProcedure

Use the `MFP.Server.invokeProcedure(invocationData)` to call other JavaScript adapters.
You can see usage examples on the Advanced Adapter Usage and Mashup (../advanced-adapter-usage-mashup) tutorial.

Logging

The JavaScript API provides logging capabilities through the MFP.Logger class. It contains four functions

that correspond to four standard logging levels.
You can see the server-side log collection (../server-side-log-collection) tutorial for more information.

JavaScript adapter examples:

Last modified on