Creating Java and JavaScript Adapters

Overview

This tutorial explains how to create MobileFirst **Java or JavaScript adapters**.
An adapter can be created either by using Maven directly or by using the MobileFirst Developer CLI (with a required prerequisite to have Maven installed on the developer workstation), or be downloaded as a sample starter code from the MobileFirst Operations Console. The Adapter code can then be edited using your IDE of choice, such as Eclipse, IntelliJ and the like.

Prerequisite: Make sure that you read the Adapters Overview (../adapters-overview) first.

Jump to:

Creating Adapters Using Maven Archetype "adapter-maven-archetype"

The "adapter-maven-archetype" is a MobileFirst-provided archetype, and is based on the Maven archetype toolkit (https://maven.apache.org/guides/introduction/introduction-to-archetypes.html) in order to create the adapter as a Maven project.

Install Maven

In order to create an adapter, you first need to download and install Maven. Go to the Apache Maven website (https://maven.apache.org/) and follow the instructions how to download and install Maven.

Create an Adapter

To create a Maven adapter project, use the `archetype:generate` command. Once the command is running, Maven will download required files from the MobileFirst Maven repository in order to be able to generate the adapter Maven project.

You can choose to run the command interactively or directly.

Interactive Mode

1. Replace the **DarchetypeArtifactId** placeholder with the actual value and run:

```
mvn archetype:generate -DarchetypeGroupId=com.ibm.mfp -DarchetypeArtifactId=<adapter type arti
fact ID> -DarchetypeVersion=8.0.0
```

- The `Archetype Group Id` and Archetype Version are required parameters to identify the archetype.
- The `Archetype Artifact Id` is a required parameter to identify the adapter type:
  - Use `adapter-maven-archetype-java` to create a Java adapter
  - Use `adapter-maven-archetype-http` to create a JavaScript HTTP adapter
  - Use `adapter-maven-archetype-sql` to create a JavaScript SQL adapter

2. Enter a Group Id (https://maven.apache.org/guides/mini/guide-naming-conventions.html) of the Maven project to be build. For example:

```
Define value for property 'groupId': : com.mycompany
```

3. Enter an Artifact Id of the Maven project **which will later be used also as the adapter name** . For example:

```
Define value for property 'artifactId': : SampleAdapter
```

4. Enter a Maven project version (the default is `1.0-SNAPSHOT`). For example:

```
Define value for property 'version':  1.0-SNAPSHOT: : 1.0
```

5. Enter an adapter package name (the default is the `groupId`). For example:

```
Define value for property 'package':  com.mycompany: : com.mypackage
```

6. Enter `y` to confirm:

```
[INFO] Using property: archetypeVersion = 8.0.0
Confirm properties configuration:
groupId: com.mycompany
artifactId: SampleAdapter
version: 1.0
package: com.mypackage
archetypeVersion: 8.0.0
 Y: : y
```

Direct Mode

Replace the placeholders with the actual values and run the command:

```
mvn archetype:generate -DarchetypeGroupId=com.ibm.mfp -DarchetypeArtifactId=<adapter type artifact ID
> -DarchetypeVersion=8.0.0 -DgroupId=<maven_project_groupid> -DartifactId=<maven_project_artifactid> -
Dversion=<maven_project_version> -Dpackage=<adapter_package_name>
```

> For more information about the `archetype:generate` command see the Maven documentation.

Dependencies

In order to use an external library in your adapter, follow one of the following suggested instructions:

Adding a local dependency:

1. Add a **lib** folder under the root Maven project folder and put the external library in it.
2. Add the library path under the `dependencies` element in the Maven project's **pom.xml** file.

For example:

```
<groupId>sample</groupId>
  <artifactId>com.sample</artifactId>
  <version>1.0</version>
  <scope>system</scope>
  <systemPath>${project.basedir}/lib/</systemPath>
</dependency>
```

Adding an external dependency:

1. Search the dependency in one of the search engines for Maven dependencies that exist online, for example The Central Repository (http://search.maven.org/).
2. Copy the POM dependency information and paste it under the `dependencies` element in the Maven project's **pom.xml** file.

The following example uses the `cloudant-client artifactId`:

```
<dependency>
  <groupId>com.cloudant</groupId>
  <artifactId>cloudant-client</artifactId>
  <version>2.2.0</version>
  <groupId>sample</groupId>
    <artifactId>com.sample</artifactId>
    <version>1.0</version>
    <scope>system</scope>
    <systemPath>${project.basedir}/lib/</systemPath>
</dependency>
```

> For more information about dependencies see the Maven documentation.

Grouping Adapters in a Single Maven Project

If you have several adapters in your project you may want to arrange them under a single Maven project. Grouping adapters provides benefits such as build all and deploy all abilities, sharing dependencies etc.

To group adapters you need to:

1. Create a root folder and call it, for example, "GroupAdapters".
2. Put the Maven adapter projects in it.

3. Create a **pom.xml** file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

   <modelVersion>4.0.0</modelVersion>
   <groupId>com.sample</groupId>
   <artifactId>GroupAdapters</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>pom</packaging>

   <modules>
        <module>Adapter1</module>
        <module>Adapter2</module>
   </modules>

   <properties>
      <!-- parameters for deploy mfpf adapter -->
      <mfpfUrl>http://localhost:9080/mfpadmin</mfpfUrl>
      <mfpfUser>admin</mfpfUser>
      <mfpfPassword>admin</mfpfPassword>
   </properties>

  <build>
      <plugins>
         <plugin>
            <groupId>com.ibm.mfp</groupId>
            <artifactId>adapter-maven-plugin</artifactId>
            <version>8.0.0</version>
            <extensions>true</extensions>
         </plugin>
      </plugins>
   </build>

</project>
```

   1. Define a `groupId` element of your choice
   2. Add an `artifactId` element - the root folder's name
   3. Add a `module` element for each adapter
   4. Add the `build` element
   5. Replace the **localhost:9080** with your MobileFirst Server IP and port.
   6. Replace the `mfpfUser` and `mfpfPassword` values with your MobileFirst admin user name and password.
4. To build or deploy all adapters, run the Maven commands from the root "GroupAdapters" project.

Creating Adapters Using MobileFirst Developer CLI

Install MobileFirst Developer CLI

Follow the installation instructions in the Downloads (file:////home/travis/build/MFPSamples/DevCenter/_site/downloads/) page to Install MobileFirst Developer

CLI.
Prerequisite: To create adapters using the Developer CLI, Maven must be installed.

Create an Adapter

To create a Maven adapter project, use the `mfpdev adapter create` command. You can choose to run the command interactively or directly.

Interactive Mode

1. Open a **Command-line** window and run:

   ```
   mfpdev adapter create
   ```

2. Enter an adapter name. For example:

   ```
   ? Enter Adapter Name: SampleAdapter
   ```

3. Select an adapter type using the arrows and the enter keys:

   ```
   ? Select Adapter Type:
     HTTP
     SQL
   ❯ Java
   ```

   - Select `HTTP` to create a JavaScript HTTP adapter
   - Select `SQL` to create a JavaScript SQL adapter
   - Select `Java` to create a Java adapter

4. Enter an adapter package (this option is valid for Java adapters only). For example:

   ```
   ? Enter Package: com.mypackage
   ```

5. Enter a Group Id (https://maven.apache.org/guides/mini/guide-naming-conventions.html) of the Maven project to be build. For example:

   ```
   ? Enter Group ID: com.mycompany
   ```

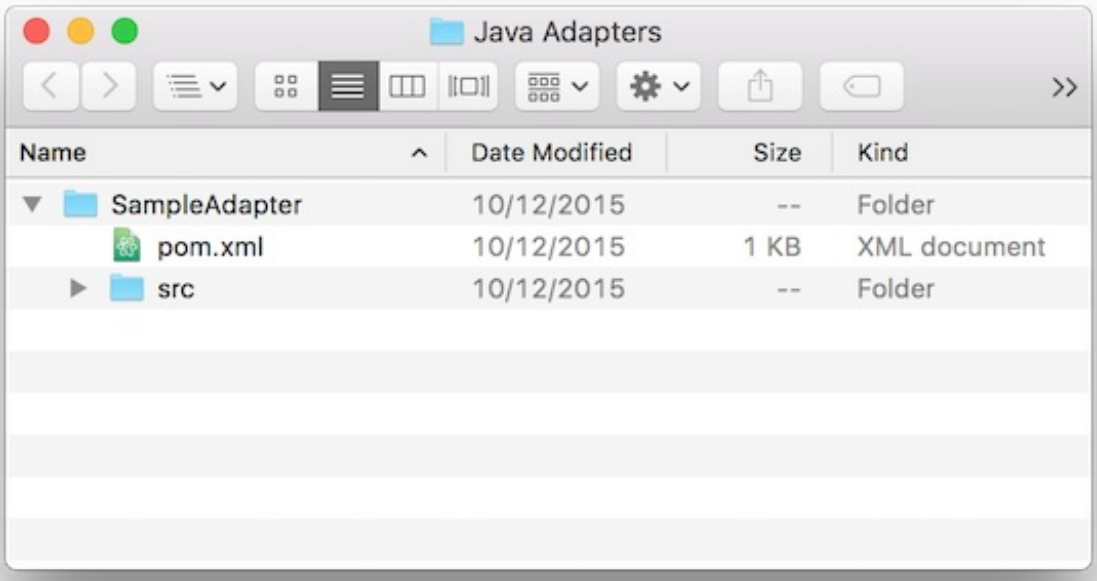
Direct Mode

Replace the placeholders with the actual values and run the command:

```
mfpdev adapter create <adapter_name> -t <adapter_type> -p <adapter_package_name> -g <maven_projec
t_groupid>
```

File Structure

After creating the adapter the result will be a Maven project containing a **src** folder and a **pom.xml** file:

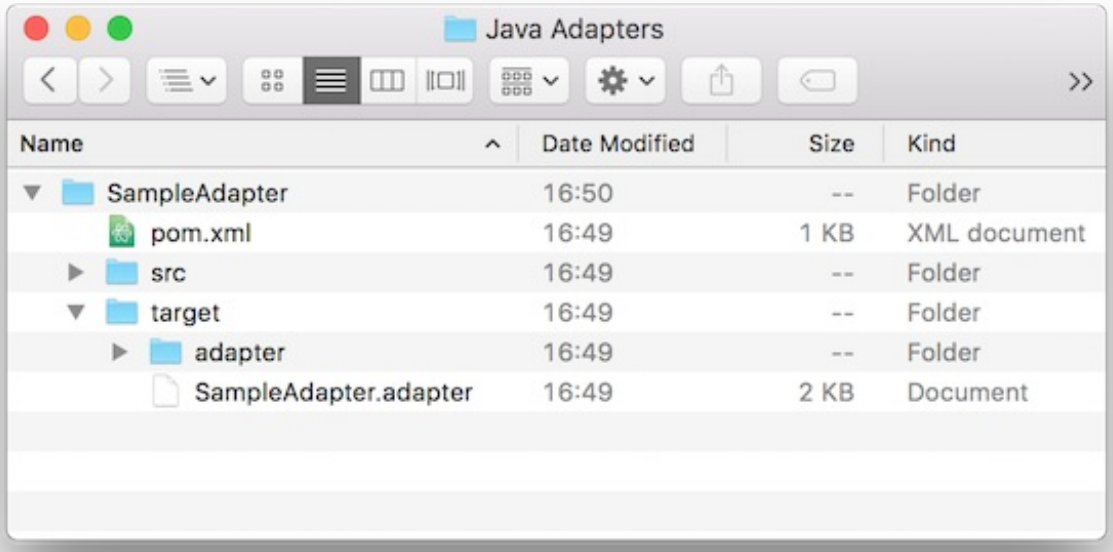


Build and Deploy Adapters

Build

- **Using the MobileFirst Developer CLI** - Run the `mfpdev adapter build` command from the project's root folder.
- **Using Maven** - The adapter is built each time you run the `mvn install` command to build the Maven project.

This generates an **.adapter** file which can be found in the **target** folder:



Deploy

1. The **pom.xml** file contains the following `properties` parameters:

```
<properties>
    <!-- parameters for deploy mfpf adapter -->
    <mfpfUrl>http://localhost:9080/mfpadmin</mfpfUrl>
    <mfpfUser>admin</mfpfUser>
    <mfpfPassword>admin</mfpfPassword>
 </properties>
```

   - Replace **localhost:9080** with your MobileFirst Server IP address and port number.
   - Replace the **mfpfUser** and **mfpfPassword** values with your MobileFirst admin user name and password.

2. Run the deploy command from the project's root folder:
   - **Using the MobileFirst Developer CLI**:

```
mfpdev adapter deploy
```

   - **Using Maven**:

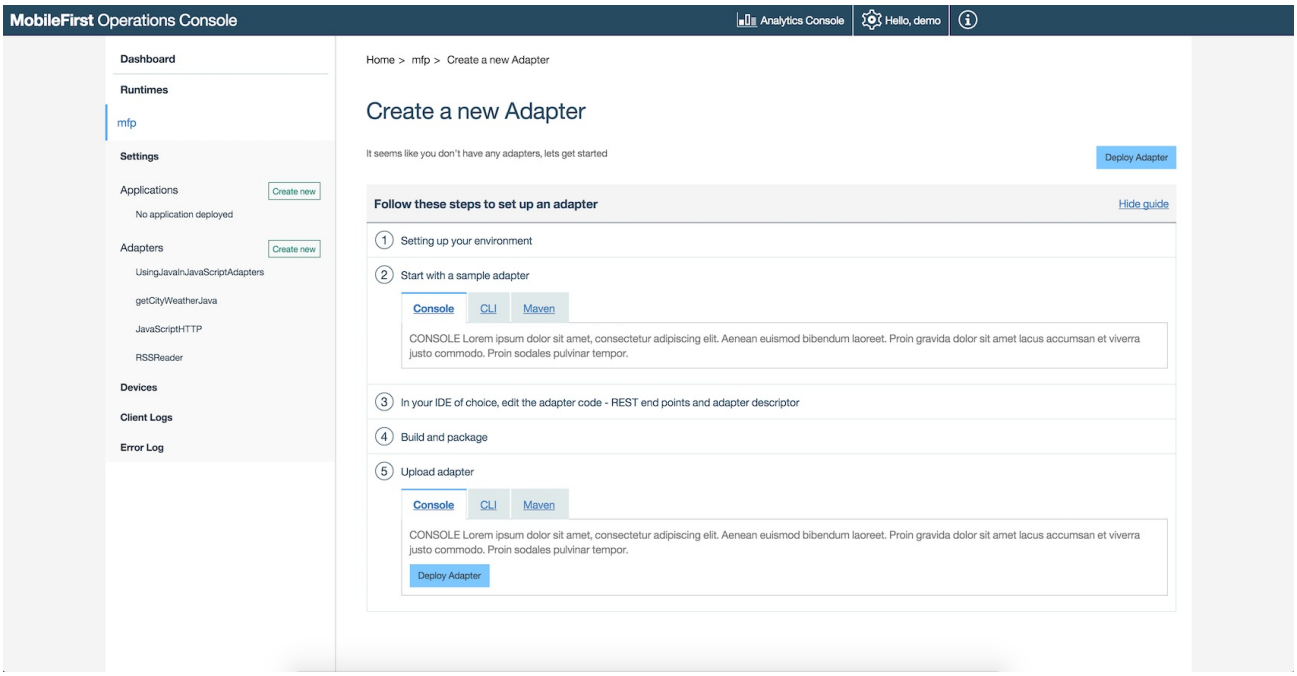
```
mvn adapter:deploy
```

> **Tip:** You can also build and deploy the adapter using a single command: `mvn install adapter:deploy`

NOTE: The deploy command is available only during development (for security reasons).

Downloading or Deploying Adapters Using MobileFirst Operations Console

1. Open your browser of choice and load the MobileFirst Operations Console using the address `http://<IP>:<PORT>/mfpconsole/`.
2. Click on the "Create new" button next to Adapters. You have two options to create an adapter:
    - Using Maven or MobileFirst Developer CLI as previously explained above.
    - Download a template adapter project (step 2).
3. Build the adapter Using Maven or MobileFirst Developer CLI.
4. Choose one of the following ways to upload the generated **.adapter** file which can be found in the target folder of the adapter project:
    - Click on the Deploy Adapter button (step 5).
    - Drag and drop the file into the Create new adapter screen.



5. After successfully deploying the adapter, the details page will be displayed containing the following tabs:
6. Configurations - properties defined by the adapter XML file. Here you can change the configurations without having to deploy again.
7. Resources - a list of the adapter resources.
8. Configurations Files - adapter configuration data, to be used in devops environments.

Tutorials to follow next

- Learn about Java adapters (../java-adapters/)
- Learn about JavaScript adapters (../javascript-adapters/)
- Develop adapters in IDEs (../developing-adapters/)
- Testing and debugging adapters (../testing-and-debugging-adapters/)
- Review all Adapters tutorials (../)