Resource Request from Native Android Applications

Overview

MobileFirst applications can access resources using the `WLResourceRequest` REST API.
The REST API works with all adapters and external resources.

Prerequisite: Ensure you have added the MobileFirst Platform SDK (../../adding-the-mfpf-sdk/adding-the-mfpf-sdk-to-android-applications) to your Native Android project.

WLResourceRequest

The `WLResourceRequest` class handles resource requests to adapters or external resources.

1. Define the URI of the resource:

   ```
   URI adapterPath = new URI("/adapters/RSSReader/getFeed");
   ```

   - For JavaScript adapters, use `/adapters/{AdapterName}/{procedureName}`
   - For Java adapters, use `/adapters/{AdapterName}/{path}`. The path depends on how you defined your @Path annotations in your Java code. This would also include any @PathParam you used.
   - To access resources outside of the project, use the full URL as per the requirements of the external server.

2. Create a `WLResourceRequest` object and choose the HTTP Method (GET, POST, etc):

   ```
   WLResourceRequest request = new WLResourceRequest(adapterPath,WLResourceRequest.GET);
   ```

3. Before sending your request, you may want to add parameters as needed.

- In JavaScript adapters, which use ordered nameless parameters, pass an array of parameters with the name `params`:

  ```
  ```js
  request.setQueryParameter("params","['param1', 'param2']");
  ```
  ```

- In Java adapters or external resources, there are several optional types of parameters:

  - **Path parameter**: path parameters (/path/value1/value2) are set during the creation of the WLResourceRequest object:

```
URI adapterPath = new URI("/adapters/JavaAdapter/users/"
        + first_name.getText().toString() +"/"
        + middle_name.getText().toString() +"/"
        + last_name.getText().toString()
);

WLResourceRequest request = new
WLResourceRequest(adapterPath,WLResourceRequest.POST);
```

- **Query parameters**: use the `.setQueryParameter` method for each parameter:

```
request.setQueryParameter("param1","value1");
request.setQueryParameter("param2","value2");
```

- **Header parameters**: use `.addHeader()` to set header parameters to the request.

```
request.addHeader("date", date.getText().toString());
```

- **Form parameters**: To send form parameters in the body, use `.send(HashMap<String, String> formParameters, WLResponseListener)`:

```
HashMap formParams = new HashMap();
formParams.put("height", height.getText().toString());
request.send(formParams, new MyInvokeListener());
```

1. Call the resource by using the `.send()` method. Specify a WLResponseListener class instance:

```
request.send(new MyInvokeListener());
```

> See the user documentation to learn more about `WLResourceRequest` and other signatures for the `send` method, which are not covered in this tutorial.

The response

When the resource call is completed, the framework calls one of the methods of the WLResponseListener class that you defined in the `.send()` method.

1. Create a new class that implements the `WLResponseListener` interface:

```
public class MyInvokeListener implements WLResponseListener {
}
```

2. Implement the `onSuccess` and `onFailure` methods.
   If the resource call is successful, the `onSuccess` method is called. Otherwise, the `onFailure` method is called.
   Use these methods to get the data that is retrieved from the adapter.
   The `response` object contains the response data and you can use its methods and properties to retrieve the required information.

```
public void onSuccess(WLResponse response) {
      String responseText = response.getResponseText();
      AndroidNativeApp.updateTextView("Successfully called the resource\n" + responseText);
}

public void onFailure(WLFailResponse response) {
     String responseText = response.getResponseText();
     AndroidNativeApp.updateTextView("Failed to call the resource\n" + responseText);
}
```

For more information

> For more information about WLResourceRequest, refer to the user documentation.

Sample application

The ResourceRequestAndroid project contains a native Android application that makes a resource request using a Java adapter.
The adapter Maven project contains the Java adapter to be used during the resource request call.

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/ResourceRequestAndroid/tree/release80) the Native project.
Click to download (https://github.com/MobileFirst-Platform-Developer-Center/Adapters/tree/release80) the adapter Maven project.

Sample usage

- Make sure to update the **app/src/main/assets/mfpclient.properties** file in the Android Studio project with the server properties.
- The sample uses the `JavaAdapter` contained in the Adapters Maven project. Use either Maven or MobileFirst Developer CLI to build and deploy the adapter (../../creating-adapters/).

SCREENSHOT