Resource request from Native Windows 8.1 Universal and Windows 10 UWP applications

Overview

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

Prerequisites:

- Ensure you have added the MobileFirst Platform SDK to your Native Windows 8.1 Universal (../../adding-the-mfpf-sdk/adding-the-mfpf-sdk-to-windows-8-applications) or Windows 10 UWP (../../adding-the-mfpf-sdk/adding-the-mfpf-sdk-to-windows-10-applications) project.
- Learn how to create adapters (../../adapters/adapters-overview/).

WLResourceRequest

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

Create a `WLResourceRequest` object and specify the path to the resource and the HTTP method.
Available methods are: `WLHttpMethodGet`, `WLHttpMethodPost`, `WLHttpMethodPut` and `WLHttpMethodDelete`.

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}`
   - To access resources outside of the project, use the full URL

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

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

3. Add the required parameters:

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

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

   - In Java adapters or external resources, use the `setQueryParameter` method for each parameter:

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

4. Call the resource by using the `.send()` method.
   Specify a `MyInvokeListener` 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 `MyInvokeListener` class.

1. Specify that the `MyInvokeListener` class implements the `WLResponseListener` interface:

   ```
   public class MyInvokeListener : 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)
   {
       WLProcedureInvocationResult invocationResponse = ((WLProcedureInvocationResult) response);
       JObject items;
       try
       {
           items = invocationResponse.getResponseJSON();
           await dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
           {
               myMainPage.AddTextToReceivedTextBlock("Response Success: " + items.ToString());
           });
       }
       catch (JsonReaderException e)
       {
            Debug.WriteLine("JSONException : " + e.Message);
       }
   }

   public void onFailure(WLFailResponse response)
   {
       await dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
       {
           myMainPage.AddTextToReceivedTextBlock("Response failed: " + response.ToString());
       });
   }
   ```

For more information

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

Sample application

The ResourceRequestWin8 and ResourceRequestWin10 projects contain a native Windows 8 Universal/Windows 10 UWP 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/ResourceRequestWin8/tree/release80) the Native project.
Click to download (https://github.com/MobileFirst-Platform-Developer-Center/ResourceRequestWin10/tree/release80) the Native project.
Click to download (https://github.com/MobileFirst-Platform-Developer-Center/Adapters/tree/release80) the adapter Maven project.

Sample usage

1. From the command line, navigate to the Visual Studio project.
2. Ensure the sample is registered in the MobileFirst Server by running the command: `mfpdev app register`.
3. The sample uses the `JavaAdapter` contained in the Adapters Maven project. Use either Maven or MobileFirst Developer CLI to build and deploy the adapter (../../adapters/creating-adapters/).
4. import the project to Visual Studio, and run the sample by clicking the **Run* button.