Resource Request from Native Windows 8 Universal Applications

Overview

MobileFirst applications can access resources using the `WLResourceRequest` REST API.
The REST API works with all adapters and external resources LINK TO using-mobilefirst-server-authenticate-external-resources.
This tutorial explains how to use the `WLResourceRequest` API with an HTTP adapter.

To create and configure a Windows 8 (Universal) native project, first follow the Adding the MobileFirst Platform Foundation SDK to Windows 8 Universal Applications (../../adding-the-mfpf-sdk/adding-the-mfpf-sdk-to-windows-8-applications) tutorial.

Calling an adapter procedure

The `WLResourceRequest` class handles resource requests to MobileFirst 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}`
   - 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","['MobileFirst_Platform']");
     ```

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

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

4. Call the procedure 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.

Receiving a procedure response

When the procedure invocation 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 procedure invocation 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());
    });
}
```

Sample application

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/InvokingAdapterProcedures) the MobileFirst project.

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/InvokingAdapterProceduresWin8) the Native project.

- The `InvokingAdapterProcedures` project contains a **MobileFirst Native API** to deploy to MobileFirst Server.
- The `InvokingAdapterProceduresWin8` project contains a **native Windows 8 Universal application** that uses a MobileFirst native API library to communicate with a MobileFirst Server instance.
- Make sure to update the `mfpclient.properties` file in **InvokingAdapterProceduresWin8** with the relevant server settings.

SCREENSHOT