Resource request from native Windows 8 Universal applications

Overview

To create and configure a Windows 8 (Universal) native project, first follow the Configuring a native Windows 8 application with the MobileFirst Platform SDK (../../hello-world/configuring-a-native-windows-8-application-with-the-mfp-sdk/) tutorial.

MobileFirst applications can access resources using the `WLResourceRequest` REST API.
This tutorial explains how to use the `WLResourceRequest` API with an HTTP adapter.

Initializing WLClient

Copy

```
WLClient client = WLClient.getInstance();
```

1. To establish a connection to MobileFirst Server, use the `connect` method by specifying the `MyConnectResponseListener` class instance as a parameter.

   Copy

   ```
   client.connect(new MyConnectResponseListener(this));
   ```

   The `WLClient` instance tries to connect to the MobileFirst Server instance according to the properties of the `wlclient.properties` file.
   After the connection is established, it invokes one of the methods of the `MyConnectResponseListener` class.

2. Specify that the `MyConnectResponseListener` class implements the `WLResponseListener` interface.

   Copy

   ```
   public class MyConnectResponseListener : WLResponseListener
   ```

   The `WLResponseListener` interface defines two methods:
   - `public void onSuccess (WLResponse response) { }`
   - `public void onFailure (WLFailResponse response) { }`

3. Use the previous methods to process connection success or connection failure.

Invoking an adapter procedure

After the connection is established with a MobileFirst Server instance, you can use the `WLResourceRequest` class to invoke adapter procedures or call any REST resources.

1. Define the URI of the resource. For a JavaScript HTTP adapter:
   `/adapters/{AdapterName}/{ProcedureName}`

   ```
   Copy

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

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

   ```
   Copy

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

3. Add the required parameters.
   - For JavaScript-based adapters, use the `params` parameter name to set an array of parameters.

     ```
     Copy

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

   - For Java adapters or other resources, you can use `setQueryParameter` for each parameter.

     ```
     Copy

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

   - Trigger the request with `.send()`.
     Specify a `MyInvokeListener` class instance as a parameter.
     You learn how to define this class instance in the next section.

     ```
     Copy

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

Receiving a procedure response

After the procedure invocation is completed, the `WLClient` instance calls one of the methods of the `MyInvokeListener` class.
As before, you must specify that the `MyInvokeListener` class implements the `WLResponseListener` interface.

```
using IBM.Worklight;
namespace InvokingAdapterProceduresWin8{
   public class MyInvokeListener : WLResponseListene
r
   { }
{
```

The `onSuccess` and `onFailure` methods are invoked by the `WLClient` instance. The response object contains the response data. 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 `wlclient.properties` file in **InvokingAdapterProceduresWin8** with the relevant server settings.