

Resource request from native Windows Phone 8 applications

Overview

To create and configure a Windows Phone 8 (Silverlight) native project, first follow the [Configuring a native Windows Phone 8 application with the MobileFirst Platform SDK \(../../hello-world/configuring-a-native-windows-phone-8-application-with-the-mfp-sdk/\)](#) 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

```
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.

```
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.

```
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 `WLClient` instance to invoke adapter procedures.

1. Create a `WLProcedureInvocationData` object with the adapter and procedure names.
2. Add the required parameters as an object array and set request options (for example: Invocation Context).

3. Get the existing `WLClient` instance and use it to invoke an adapter procedure.
4. Specify the `MyInvokeListener` class instance as a parameter.

```
WLProcedureInvocationData invocationData = new WLProcedureInvocationData("RSSReader",
"getFeed");
invocationData.setParameters(new Object[]{});
String myContextObject = "InvokingAdapterProceduresWP8";
WLRequestOptions options = new WLRequestOptions();
options.setInvocationContext(myContextObject);
WLClient.getInstance().invokeProcedure(invocationData, new MyInvokeListener(this), options);
```

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 InvokingAdapterProceduresWP8{
    public class MyInvokeListener :
        WLResponseListener
    {
        {}
    }
}
```

The `onSuccess` and `onFailure` methods are invoked by the `WLClient`. 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();
        Deployment.Current.Dispatcher.BeginInvoke() =>
        {
            myMainPage.AddTextToReceivedTextBlock("Response Success: " + items.ToString());
        });
    }
    catch (JsonReaderException e)
    {
        Deployment.Current.Dispatcher.BeginInvoke() =>
        {
            myMainPage.AddTextToReceivedTextBlock("JSONException : " + e.Message);
        });
    }
}

public void onFailure(WLFailResponse response)
{
    Deployment.Current.Dispatcher.BeginInvoke() =>
    {
        myMainPage.AddTextToReceivedTextBlock("Response failed: " + response.ToString());
    });
}

```

Sample application

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/InvokingAdapterProcedures/tree/release71>) the MobileFirst project.

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/InvokingAdapterProceduresWP8/tree/release71>) the Native project.

- The `InvokingAdapterProcedures` project contains a **MobileFirst Native API** to deploy to MobileFirst Server.
- The `InvokingAdapterProcedures` project contains a **native Windows Phone 8 application** that uses a MobileFirst native API library to communicate with a MobileFirst Server instance.

Make sure to update the `wlclient.properties` file in `NativeWP8Invoking` with the relevant server settings.

