# Resource request from JavaScript (Cordova, Web) applications

## Overview

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

**Prerequisites**:

- If you are implementing a Cordova application, ensure you have added the MobileFirst Platform SDK (../../../adding-the-mfpf-sdk/cordova) to your Cordova application.
- If you are implementing a Web application, ensure you have added the MobileFirst Platform SDK (../../../adding-the-mfpf-sdk/web) to your Web application.
- Learn how to create adapters (../../../adapters/creating-adapters/).

## 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: `WLResourceRequest.GET`, `WLResourceRequest.POST`, `WLResourceRequest.PUT` and `WLResourceRequest.DELETE`.

```
var resourceRequest = new WLResourceRequest(
    "/adapters/JavaAdapter/users",
    WLResourceRequest.GET
);
```

- 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.
- **timeout**: Optional, request timeout in milliseconds

## Sending the request

Request the resource by using the `send()` method.
The `send()` method takes an optional parameter to set a body to the HTTP request, which could be a JSON object or a simple string.

Using JavaScript **promises**, you can define `onSuccess` and `onFailure` callback functions.

```
resourceRequest.send().then(
    onSuccess,
    onFailure
)
```

### setQueryParameter

By using the `setQueryParameter` method, you can include query (URL) parameters in the REST request.

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

#### JavaScript adapters

JavaScript adapters use ordered nameless parameters. To pass parameters to a Javascript adapter, set an array of parameters with the name `params`:

```
resourceRequest.setQueryParameter("params", "['value1', 'value2']");
```

This should be used with `WLResourceRequest.GET`.

### setHeader

By using the `setHeader` method, you can set a new HTTP header or replace an existing header with the same name in the REST request.

```
resourceRequest.setHeader("Header-Name","value");
```

### sendFormParameters(json)

To send URL-encoded form parameters, use the `sendFormParameters(json)` method instead. This method converts the JSON to a URL encoded string, sets the `content-type` to `application/x-www-form-urlencoded`, and sets it as the HTTP body:

```
var formParams = {"param1": "value1", "param2": "value2"};
resourceRequest.sendFormParameters(formParams);
```

#### JavaScript adapters

JavaScript adapters use ordered nameless parameters. To pass parameters to a Javascript adapter, set an array of parameters with the name `params`:

```
var formParams = {"params":"['value1', 'value2']"};
```

This should be used with `WLResourceRequest.POST`.

> For more information about `WLResourceRequest`, see the API reference in the user documentation.

## The response

Both the `onSuccess` and `onFailure` callbacks receive a `response` object. The `response` object contains the response data and you can use its properties to retrieve the required information. Commonly used properties are `responseText`, `responseJSON` (JSON object, if the response is in JSON) and `status` (the HTTP status of the response).

In case of request failure, the `response` object also cotains a `errorMsg` property.
Depending if using a Java or JavaScript adapter, the response may contain other properties such as `responseHeaders`, `responseTime`, `statusCode`, `statusReason`, and `totalTime`.


```
{
  "responseHeaders": {
    "Content-Type": "application/json",
    "X-Powered-By": "Servlet/3.1",
    "Content-Length": "86",
    "Date": "Mon, 15 Feb 2016 21:12:08 GMT"
  },
  "status": 200,
  "responseText": "{\"height\":\"184\",\"last\":\"Doe\",\"Date\":\"1984-12-12\",\"age\":31,\"middle\":\"C\",\"first
\":\"John\"}",
  "responseJSON": {
    "height": "184",
    "last": "Doe",
    "Date": "1984-12-12",
    "age": 31,
    "middle": "C",
    "first": "John"
  },
  "invocationContext": null
}
```


### Handling the response

The response object is received by the `onSuccess` and `onFailure` callback functions.
For example:

```
onSuccess: function(response) {
    resultText = "Successfully called the resource: " + response.responseText;
},

onFailure: function(response) {
    resultText = "Failed to call the resource:" + response.errorMsg;
}
```

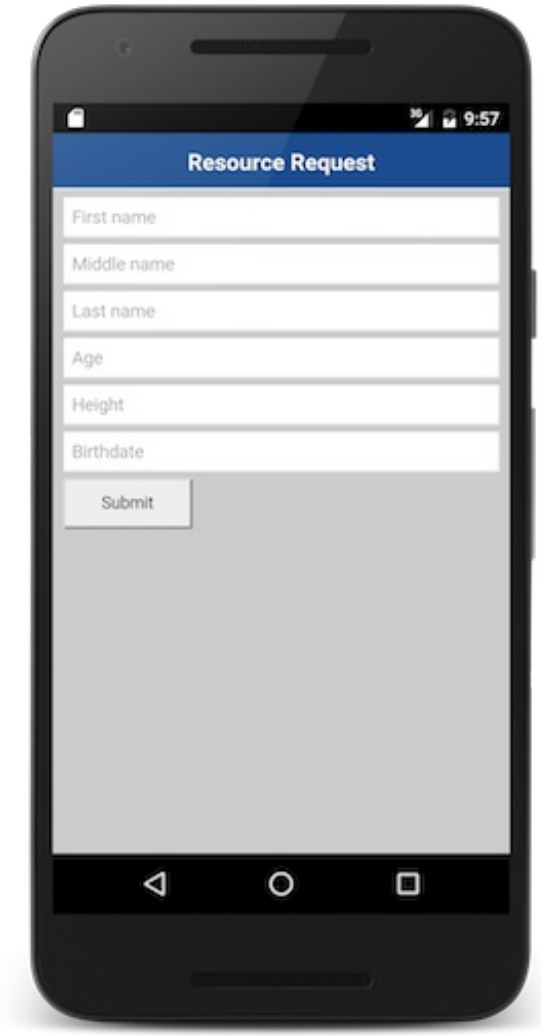
## For more information

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

# Sample applications

The **ResourceRequestWeb** and **ResourceRequestCordova** projects demonstrate a resource request using a Java adapter.
The adapter Maven project contains the Java adapter used during the resource request call.

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/ResourceRequestCordova/tree/release80) the Cordova project.
Click to download (https://github.com/MobileFirst-Platform-Developer-Center/ResourceRequestWeb/tree/release80) the Web project.
Click to download (https://github.com/MobileFirst-Platform-Developer-Center/Adapters/tree/release80) the adapter Maven project.




## Web sample usage

Make sure you have Node.js installed.

1. Register the application in the MobileFirst Operations Console.
2. Start the reverse proxy by running the commands: `npm install` followed by: `npm start`.
3. In a browser, load the URL http://localhost:9081/sampleapp (http://localhost:9081/sampleapp).

## Sample usage

Follow the sample's README.md file for instructions.