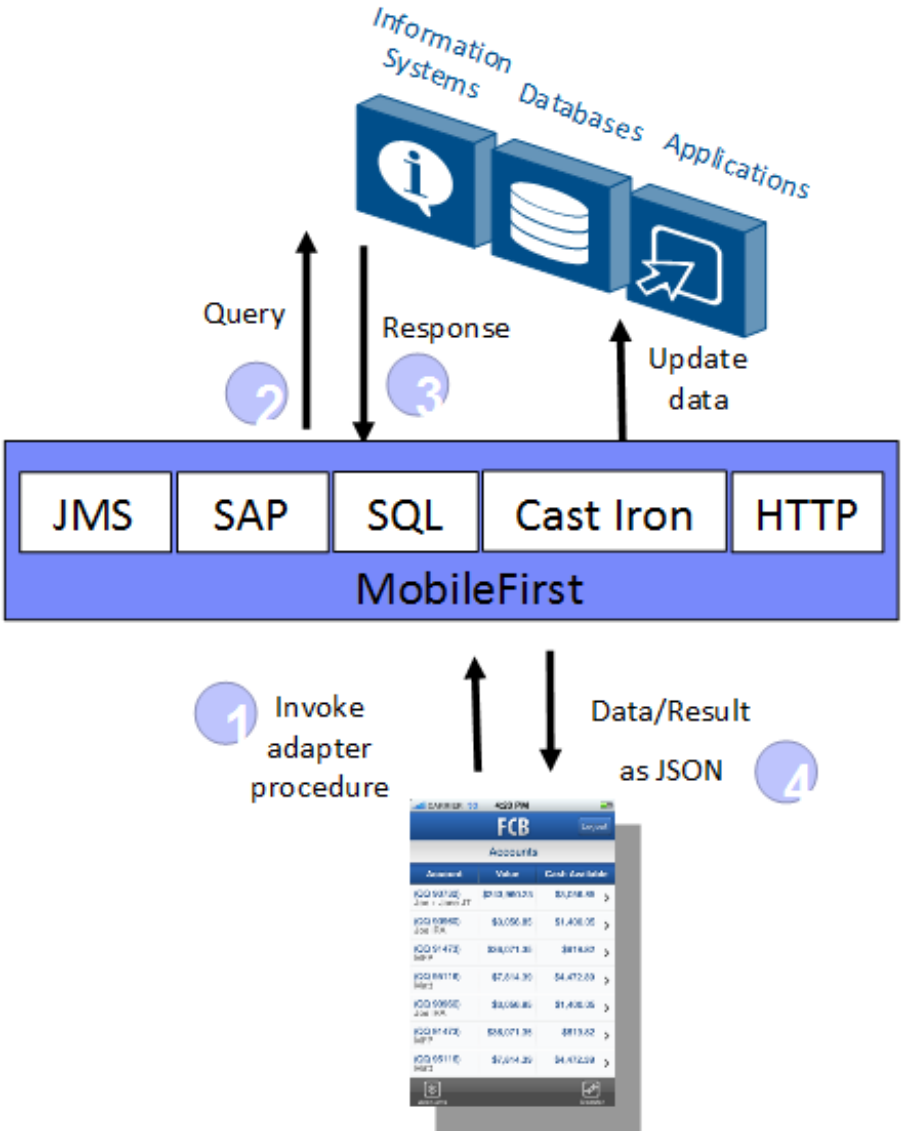
# Adapters overview

## Overview

This tutorial presents how to develop server-side code (adapters) that is used for integrating between the client application, through the MobileFirst Platform, with enterprise back-end applications and cloud services.

The tutorial is followed by explanations and examples for HTTP, SQL, Cast Iron, and JMS adapters. It also shows how to invoke adapter procedures from hybrid and native applications, advanced adapter usage, and how to use Java in adapters.




## Benefits

### Universality

- Adapters support multiple integration technologies and back-end information systems.

### Read-only and transactional capabilities

- Adapters support read-only and transactional access modes to back-end systems.

### Fast development

- Adapters use simple XML syntax and are easily configured with JavaScript API.

### Security

- Adapters use flexible authentication facilities to create connections with back-end systems.
- Adapters offer control over the identity of the connected user.

### Transparency

- Data that is retrieved from back-end applications is exposed in a uniform manner, regardless of the adapter type.

## Adapters anatomy

Each adapter consists of:

- An XML file, which describes the connectivity options and lists the procedures that are exposed to the application or other adapters.
- A JavaScript file, which contains the implementation of procedures that are declared in the XML file.
- Zero, one or more XSL files, which contain a transformation scheme for retrieved raw XML data.

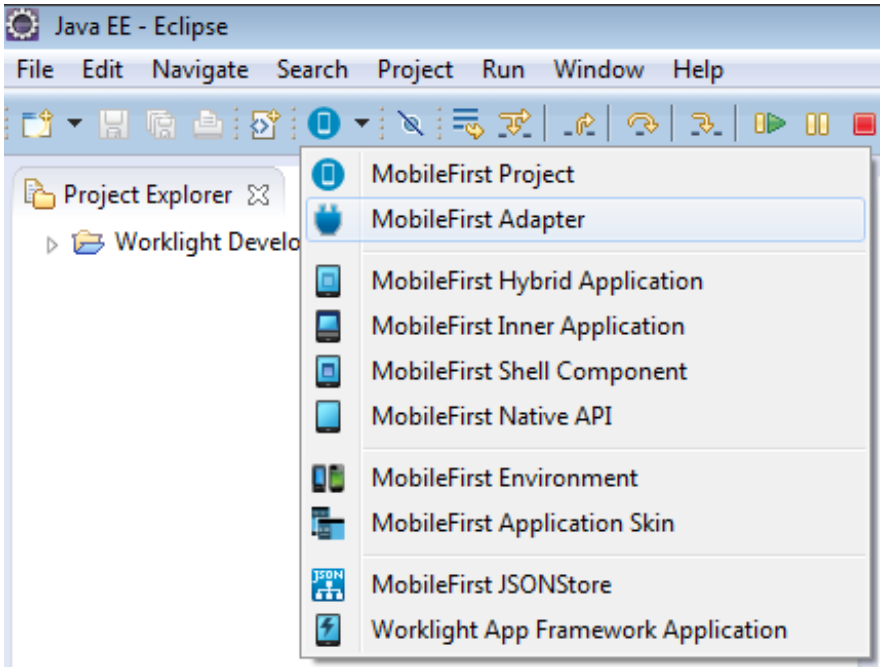
Data that is retrieved by an adapter can be returned raw or preprocessed by the adapter itself. In either case, it is presented to the application as a **JSON object**.

## Adapter procedures anatomy

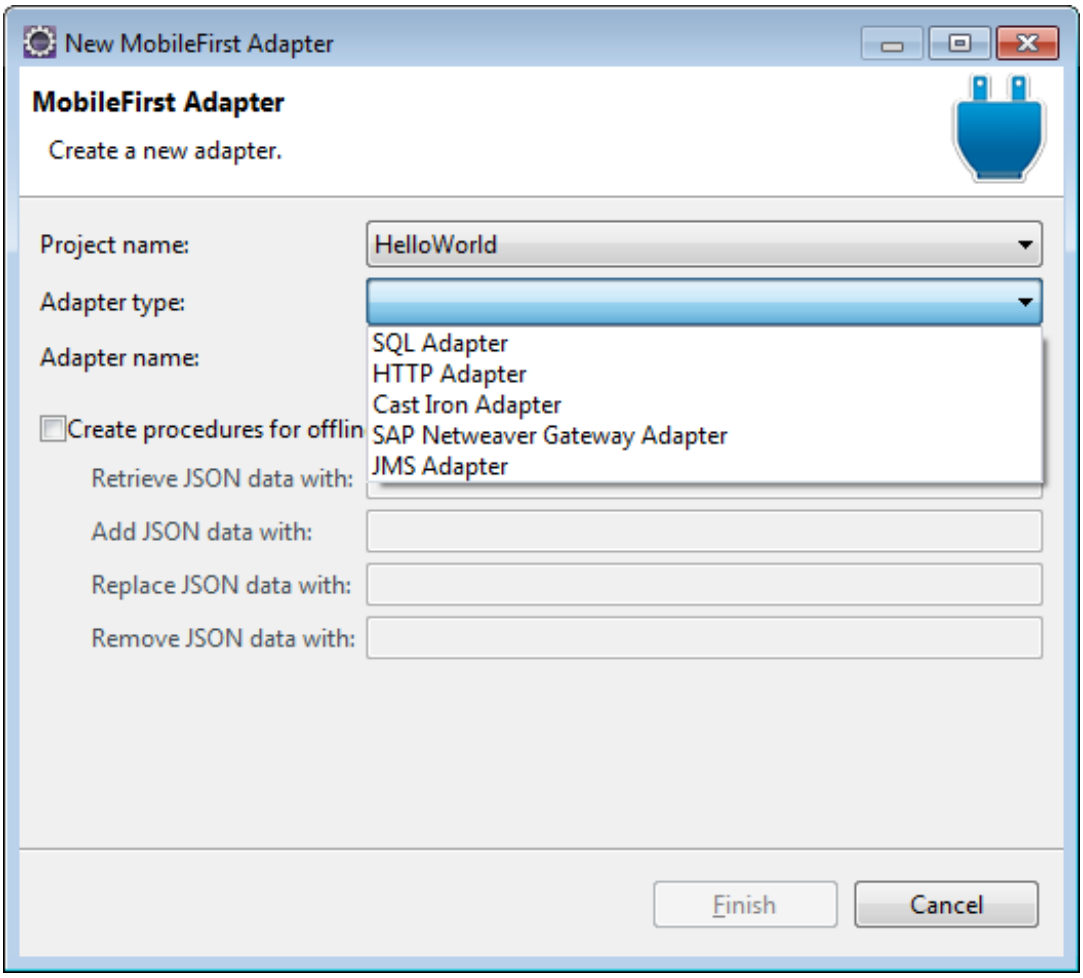
Procedures provide adapter functions to the application.
Procedures call back-end services to retrieve data or to perform actions.
Procedures are declared in XML and are implemented with server-side JavaScript.
By using server-side JavaScript, a procedure can process the data before or after it calls the service.
More filtering can be applied to retrieved data with simple XSLT.
Usually, adapter procedures are implemented in JavaScript, but because an adapter is a server-side entity, it is possible to use Java in the adapter code.

## Adapter creation

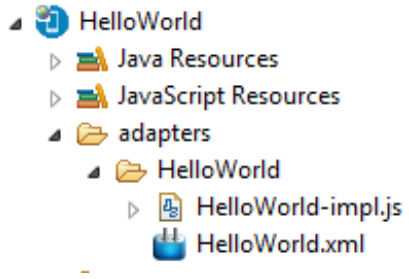
1. In Eclipse, click the MobileFirst icon that is located in the toolbar and select **MobileFirst Adapter**.




2. Select a MobileFirst project and an adapter type.



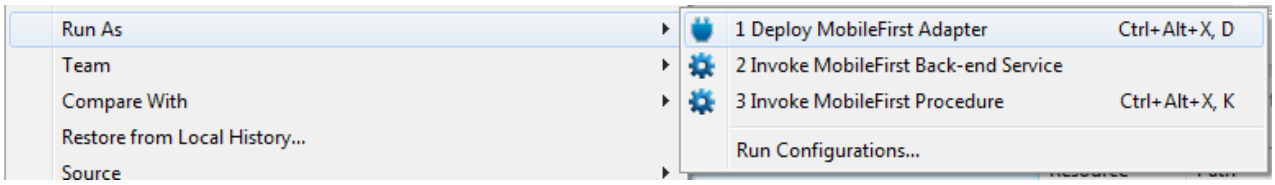

3. Select an adapter type and type an adapter name. Applications use this name to access the adapter.
4. Click **Finish**.



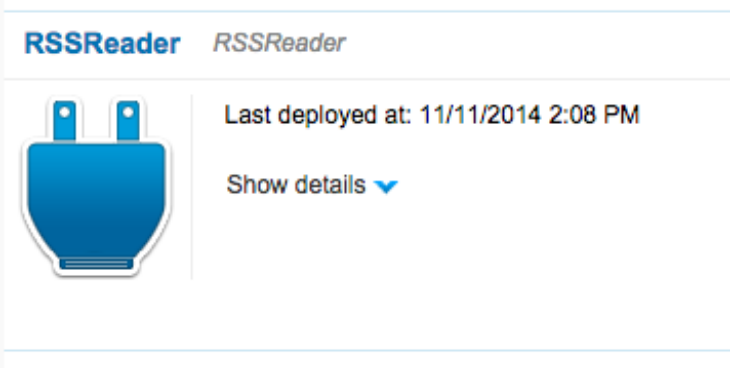

This generates both an XML file, which declares procedures and connection properties, and a JavaScript file, which defines procedures and the adapter logic.

## Adapter deployment

1. Select an adapter to deploy.
2. Right-click the adapter and select **Run As > Deploy MobileFirst Adapter**.



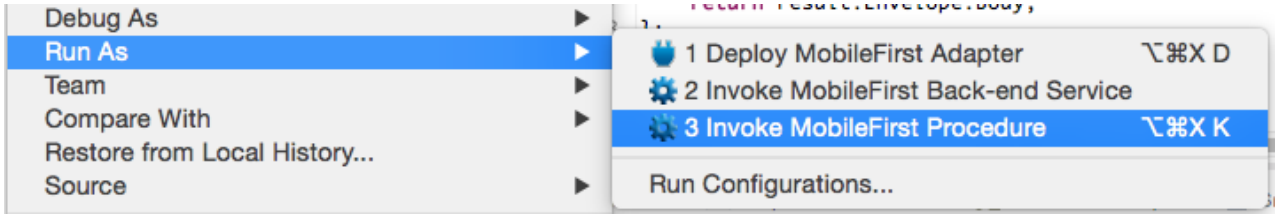

MobileFirst Studio archives the adapter code and deploys it to the MobileFirst Server instance.
You can see the deployed adapter in the MobileFirst Console.



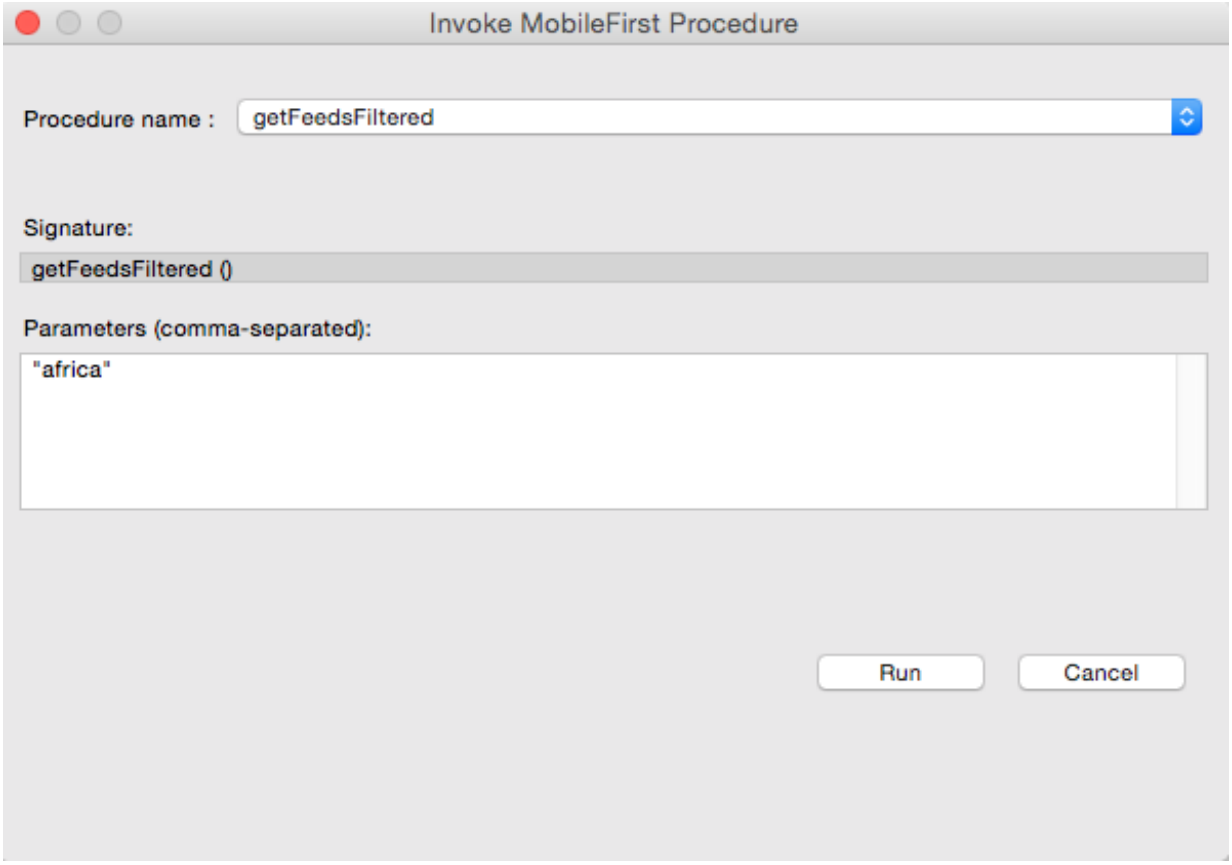

## Testing adapter procedures

MobileFirst Studio can be used to test the adapter procedures.
To run a procedure test:

1. Select **Run As > Invoke MobileFirst Procedure**.



2. Select the procedure that you want to test.
3. Enter comma-separated procedure parameters and click **Run**.




## Adapter XML structure

```
<wl:adapter name="HelloWorld">
 <displayName />
 <description />
 <connectivity>
  <connectionPolicy>
  <loadConstraints>
 </connectivity>
 <procedure />
 <procedure />
 ...
</wl:adapter>
```

- `name: Mandatory. The name of the adapter.`
- `displayName`: Optional. The name that is displayed in MobileFirst Console
- `description`: Optional. Additional information that is displayed in the MobileFirst Console
- `connectivity`
  - Defines the connection properties and load constraints of the back-end system.
  - When the back-end system requires user authentication, defines how user credentials are obtained.
- `procedure`: Declares a service for accessing a back-end application. One entry for each adapter

procedure.

```
<procedure name="procedure1"></procedure>
<procedure name="procedure2"></procedure>
```

## Adapter JavaScript structure

Each procedure declared in the adapter XML file must have a corresponding function in the JavaScript file. `WL.Server` API defines a procedure logic in JavaScript.

```
function procedure1(param) {
  return WL.Server.invokeSQLStatement({
    preparedStatement: procedure1Statement
,
    parameters: [param]
  });
}
```